Cloud Gaming: Understanding the Support From Advanced Virtualization and Hardware

Ryan Shea, *Student Member, IEEE*, Di Fu, *Student Member, IEEE*, and Jiangchuan Liu, *Senior Member, IEEE*

***Abstract*— Existing cloud gaming platforms have mainly focused on private nonvirtualized environments with proprietary hardware. Modern public cloud platforms heavily rely on *virtualization* for efficient resource sharing, the potentials of which have yet to be explored. Migrating gaming to a public cloud is nontrivial, however, particularly considering the overhead for virtualization and that the graphics processing units (GPUs) for game rendering has long been an obstacle in virtualization. This paper takes a first step toward bridging the online gaming system and the public cloud platforms. We present the design and implementation of a fully virtualized cloud gaming platform with the latest hardware support for both remote servers and local clients. We explore many critical design issues inherent in cloud gaming, including the choice of hardware or software video encoding, and the configuration and the detailed power consumption of thin client. We demonstrate that with the latest hardware and virtualization support, gaming over virtualized cloud can be made possible with careful optimization and integration of the different modules. We also highlight critical challenges toward full-fledged deployment of gaming services over the public virtualized cloud.**

***Index Terms*— Cloud computing, graphics processing unit (GPU), video gaming, virtualization.**

I. Introduction

FUELED by elastic resource provisioning, reduced costs and unparalleled scalability, *cloud computing* is drastically changing the operation and business models of the IT industry [1]. A wide range of conventional applications, from file sharing and document synchronization to media streaming, have experienced a great leap forward in terms of system efficiency and usability through leveraging cloud resources with *computation offloading*. Recently, advances in cloud technology have expanded to facilitate offloading more complex tasks such as high-definition 3-D rendering, which turns the idea of *cloud gaming* into a reality. Cloud gaming, in its simplest form, renders an interactive gaming application remotely in the cloud and streams the scenes as a video sequence back to the player over the Internet. A cloud gaming player interacts with the application through a *thin client*, which is responsible for displaying the video from the cloud rendering server as well as collecting the player's commands and sending the interactions back to the cloud.

This new paradigm of gaming services brings immense benefits by expanding the user base to the vast number of less powerful devices that support thin clients only, particularly smartphones and tablets [2]. Extensive studies have explored the potential of the cloud for gaming and addressed challenges therein [3]–[5]. Open-source cloud gaming systems such as *GamingAnywhere* for Android OS [6] have been developed. We have also seen industrial deployment, e.g., OnLive [7] and Gaikai [8]. The U.S. $380 million purchase of Gaikai by Sony [9], an industrial giant in digital entertainment and consumer electronics, and the forthcoming integration of Gaikai and Sony's Play Station 4 show that cloud gaming is beginning to move into the mainstream.

These existing cloud gaming platforms tend to focus on private nonvirtualized environments with proprietary hardware, where each user is mapped in a one-to-one fashion to a physical machine in the cloud. Modern public cloud platforms heavily rely on *virtualization*, which allows multiple *virtual machines* to share the underlying physical resources, making truly-scalable *play-as-you-go* service possible. Despite the simplicity and ease of deployment, existing cloud gaming platforms have yet to unleash the full potentials of the modern cloud toward the expansion to ultralarge scale with flexible services.

Migrating gaming to a public cloud, e.g., Amazon EC2, is nontrivial, however. The system modules should be carefully planned for effective virtual resource sharing with minimum overhead. Moreover, as the complexity of 3-D rendering increases, modern game engines not only rely on the general purpose CPU for computation but also on dedicated *graphics processing units* (GPUs). While GPU cards have been virtualized to some degree in modern virtualization systems, their performance has historically been poor for the given ultrahigh memory transfer demand and the unique data flows [10]. Recent advances in terms of both hardware and software design have not only increased the usability and performance of GPUs but also created new classes of GPUs specifically for virtualized environments. A representative is NVIDIA's recently released GRID Class GPUs, which allows multiple virtualized systems to each utilize a dedicated GPU by placing several logical GPUs on the same physical GPU board. It also contains a hardware H.264 video encoder and similar onboard hardware encoders are available in GPUs from other industry leaders such as Intel's Quick Sync Video and Advanced Micro Devices (AMD's) video coding engine.

Manuscript received September 29, 2014; revised February 7, 2015 and April 16, 2015; accepted June 12, 2015. Date of publication June 26, 2015; date of current version December 3, 2015. This paper was recommended by Associate Editor M. Abdallah.

The authors are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: rws1@cs.sfu.ca; dif@cs.sfu.ca; jcliu@cs.sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2450172

These new hardware advances from nearly every major GPU vendor allow us to take a step forward in the deployment of online gaming systems in a public cloud environment.

This paper takes a first step toward bridging the online gaming system and the public cloud platforms. We present a systematic study of many critical aspects of migrating gaming services to a virtualized cloud environment. We first closely examine the technology evolution of GPU virtualization and pass through and measure the performance of both the earlier and the advanced solutions available in the market. We then present the design and implementation of a fully virtualized cloud gaming platform, Rhizome, with the latest hardware support for both remote servers and local clients. Based on this platform, we explore many critical design issues inherent in cloud gaming, including the choice of hardware or software video encoding, and the configuration and the detailed power consumption of thin client. We demonstrate that with the latest hardware and virtualization support, gaming over a virtualized cloud can be made possible with careful optimization and integration of the different modules. We also highlight critical challenges toward full-fledged deployment of gaming services over public virtualized cloud.

II. Background and Related Work

A. Gaming Over Cloud

Using remote servers and, more recently, cloud computing for gaming has attracted significant interest. Winter *et al.* [11] designed a framework to adopt thin-client streaming and interactive games. This hybrid system rendered certain scenes locally at the thin client and thus greatly reduces the bandwidth requirements on the clients. To better understand the performance of thin-client gaming applications, Chang *et al.* [12] proposed a measurement approach to learn the implementations of thin-client games even when they are closed sourced. A follow-up study from Lee *et al.* [13] further investigated the latency issues of playing thin-client games with different network configurations.

To understand the user-perceived quality of experience in cloud gaming systems, Jarschel *et al.* [14] conducted a subjective study, in which the selected participants were asked to play slow, medium, and fast video games under different latency and packet-loss conditions. Hemmati *et al.* [15] examined the player's experience of cloud gaming with selective object encoding methods. A recent study from Claypool *et al.* [4] provided a detailed measurement on OnLive [7], a commercially available cloud gaming system, and closely analyzed its bitrates, packet sizes, and inter-packet times for both upstream and downstream game traffic. To further clarify its streaming quality, Shea *et al.* [3] measured the real-world performance of OnLive under different types of network and bandwidth conditions, revealing critical challenges toward the widespread deployment of cloud gaming. Wu *et al.* [16] further conducted a series of passive measurements on a large-scale cloud gaming platform and identified the performance issues of queuing delay as well as response delay among users. Cai *et al.* [2] suggested that the existing cloud service providers could also offer gaming as a service in their business models. Further, Chen *et al.* [17] and Vankeirsbilck *et al.* [18] have proposed measurement suites to evaluate both subject and objective QoS of interactive cloud applications including cloud gaming. Work has also been done on analyzing and optimizing clouds for game deployments [19]–[21].

Investigating the commercial cloud gaming systems, Lee *et al.* [5] proposed a system design that delivers real-time gaming interactivity as fast as traditional local client-side execution, despite the network latencies. Huang *et al.* [6], [22] provided an open-source cloud gaming system, *GamingAnywhere*, which has been deployed on the Android OS with extensive experiments performed. Recently, the network traffic of thin clients for both remote-desktop and cloud gaming applications have been carefully analyzed [23], [24]. These studies have mainly focused on private nonvirtualized cloud environments with proprietary hardware. The potentials of advanced hardware for both thin clients and public cloud with resource virtualization have yet to be understood.

B. Virtualization Techniques

Resource virtualization is a key building block of the modern public cloud with elastic service provisioning for massive users. There are several virtualization techniques pertinent to this paper, the first of which is *platform virtualization* that focuses on creating virtual machines and keeping them running in an isolated fashion. It has two major versions, namely, paravirtualization machine (PVM) and hardware virtual machine (HVM). The other techniques are *GPU virtualization* and *GPU pass-through*, aiming to achieve legitimate utilization of GPU in a virtualized environment.

1) Platform Virtualization: PVM is one of the first adopted versions of platform virtualization and is still widely deployed today. It requires special kernels and drivers to send privileged calls, which means PVM must use a modified OS to work with the hypervisor. On the other hand, HVM eliminates the needs of OS modification. It employs special hardware extensions to trap privileged calls from guest virtual machine (VMs). However, HVMs can also have higher virtualization overhead than PVM and as such may not always be the best choice for a particular situation [25]. Yet such overhead can be alleviated using paravirtualization I/O drivers. Using PVM and HVM, the CPU and many of the computer resources have been well virtualized, laying a solid foundation for today's cloud computing. Typical PVM solutions include Xen [26] and User Mode Linux. Amazon, the current industry leader in cloud computing, uses Xen to power its EC2 platform [27].

2) GPU Virtualization and Pass-Through: Initial attempts have been made to virtualize GPU, e.g., in VMWare [10]. Yet, full virtualization of GPU remains difficult, because the GPU possesses a fundamentally different architecture from CPU. GPU excels at data-parallel computations but falls behind the task-parallel ones, and thus allowing multiple VM to share the same GPU without considerable performance loss is still challenging. Fortunately, recent hardware advances have enabled virtualization systems to perform a one-to-one mapping between a device and a virtual machine guest, allowing hardware devices that do not virtualize well to be still used by a VM, including GPU. A single VM can now have a dedicated hardware mapping with the GPU, and this is known as the GPU pass through. With this technique, cloud platforms are able offer virtual machine instances with GPU capabilities. For example, Amazon EC2 has added a new instance class known as the `GPU Instances`, which has advanced pass-through enabled NVIDIA GPUs for graphics and general-purpose GPU computing.

There have been recent studies on enabling multiple VMs to access compute unified device architecture (CUDA)-enabled GPUs [28], [29], analyzing the performance of CUDA applications using a GPU pass-through device in Xen [30], as well as GPU resource scheduling in cloud [31], [32]. Despite these pioneering efforts, the suitability of modern implementations of GPU pass-through devices for cloud gaming remains to be explored fully.

III. Advanced GPU Pass-Through and Gaming Performance: A Reality Check

To understand the performance improvement and whether the technology is ready for gaming over public cloud, we have conducted a series of experiments over both an earlier GPU pass-through platform and the latest platform. With direct access to the hardware, these local virtualized platforms facilitate the measurement of virtualization overhead on game applications. We are also able to measure energy consumption of our local platform, which we are not able to obtain from a public cloud platform. As such, our tests cover a broad spectrum of metrics, including frame rate, energy consumption, and CPU memory bandwidth, both with a single user exclusively using the entire resources and with multiple users sharing resources. We will now compare an older more primitive implementation of virtualized device pass through from 2011 to a newer more optimized version from 2014. For brevity, we will refer to these two platforms as earlier and advanced, respectively. Since we are interested in the impact virtualization, overhead has on gaming performance, we also compare these systems with their optimal nonvirtualized performance (referred to as bare-metal performance).

A. Earlier GPU Pass-Through Platform (2011)

Our first test system is a server with an AMD Phenom II 1045t six-core processor running at 2.7 GHz. The motherboard's chipset is based on AMD's 990X, and we enabled AMD-V and AMD-Vi in the basic input/output system (BIOS) as they are required for HVMs' support and device pass through. The server is equipped with 16 GB of an 1333-MHz DDR-3 synchronous dynamic random access memory and the physical network interface is a 1000-Mbit/s Broadcom Ethernet adapter attached to the peripheral component interconnect express (PCI-E) bus. The GPU is an AMD-based Sapphire HD 5830 Xtreme with 1 GB of GDDR5 memory.

The Xen 4.0 hypervisor is installed on our test system, and the host and VM guests used Debian as their operating system. We configure Xen to use the HVM mode, since the GPU pass through requires hardware virtualization extensions. The VM is given access to six virtual CPU (VCPUs) and 8048 MB of RAM.

B. Advanced GPU Pass-Through Platform (2014)

Our second system is a state-of-the-art server with an Intel Haswell Xeon E3-1245 quad-core (eight threads) processor. The motherboard utilizes Intel's C226 chipset, which is one of Intel's latest server chipsets, supporting device pass through using VT-D. The server has 16 GB of a 1600-MHz error-correcting code DDR-3 memory installed. Networking is provided through an Intel i217LM 1000-Mbit/s Ethernet card. We have also installed an AMD-based Sapphire R9-280x GPU with 3 GB of GDDR5 memory, which is representative of advanced GPUs in the market. The Xen 4.1 hypervisor is installed and the VM guests again use Debian as their operating system. On this basis, we configure Xen to use the HVM mode, and the VM is given access to eight VCPUs and 8048 MB of RAM. Since our system's physical hardware supports Intel's hyper-threading technology, we enable it in the BIOS for both virtualized and bare-metal experiments.

C. Comparison and Benchmarks

As for the comparison of the optimal baseline, for both systems, we run each test on a bare-metal setup with no virtualization, i.e., the system has direct access to the hardware. The same drivers, packages, and kernel were used as in the previous setup. This particular configuration enabled us to calculate the amount of performance degradation that a virtualized system can experience.

To determine the overall system's power consumption, referred to as *wall power*, we have wired a digital multimeter (Mastech MAS-345) into the ac input power line of our system. The meter has an accuracy rating of $\pm 1.5\%$. We read the data from our meter using a data logger PCLink installed on a workstation and collect samples every second throughout our experiments. Both systems are powered by 750-W 80 PLUS Gold power supplies with active power factor correction.

To compare the pass-through performance, we have selected two game engines, both of which have cross-platform implementation, and can run natively on our Debian Linux machines. The first is `Doom 3`, which is a popular game released in 2005 and utilizes OpenGL to provide high-quality graphics. The second is the Unigine's `Sanctuary` benchmark,[1] which is an advanced benchmarking tool that runs on both Windows and Linux. The Unigine engine uses the latest OpenGL hardware to provide rich graphics that are critical to many state-of-the-art games. For each of the following experiments, we run each benchmark three times and depict the average. For Doom3 and Sanctuary, we give the results in *frames per second*.

D. Frame Rate and Energy Consumption

For Doom 3, we use the ultrahigh graphics settings with $8\times$ antialiasing (AA) and $8\times$ anisotropic filtering (AF), with a display resolution of 1920×1080 pixels (1080p). To perform the actual benchmark, we utilize Doom 3's built-in time demo, which loads a predefined sequence of game frames as fast as the system will render them.

[1] https://unigine.com/products/sanctuary/

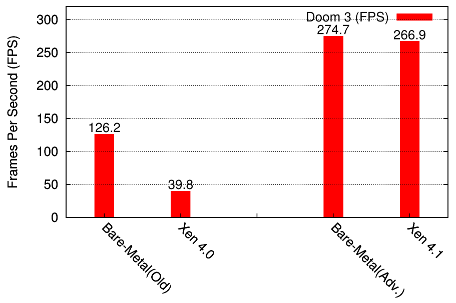


Fig. 1. Doom 3 performance.

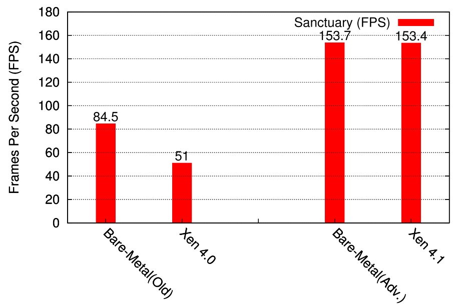


Fig. 2. Unigine sanctuary performance.

TABLE I
POWER CONSUMPTION: GAMES' BARE-METAL VERSUS VIRTUAL MACHINE

	Bare-metal	Virtual Machine	Energy
Doom 3 (Joules) Old	3696.9 J	11722.3 J	217%
Unigine Sanctuary (Watts)	265.2 W	273.6 W	3.2%
Doom 3 (Joules) Adv.	1593.9 J	1676.7 J	5.2%
Unigine Sanctuary (Watts)	289.2 W	301.7 W	4.4%

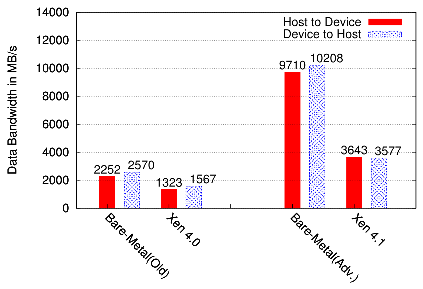


Fig. 3. Memory bandwidth by the system.

In Fig. 1, we show the results of frame rates running on our bare metal systems as well as on the Xen virtualized systems. We start the discussion with our older 2011 server. This bare-metal system performs at 126.2 frames/s, while our virtualized system dramatically falls over 65% to 39.2 frames/s. The advanced system of 2014 processes the frames at over 274 frames/s when run directly on the hardware; when run inside a virtual machine, it fails less than 3%.

Table I gives the energy consumption experienced by our machines in this experiment, measured in joules (since each system has a different running time in the test). The older system consumes over twice the amount of energy to perform the Doom 3 time demo when the system is virtualized. The advanced platform fares much better with virtualization, only adding 5.4% to energy consumption. This first virtualization experiment makes it clear that the device pass-through technology has come a long way in terms of performance and overhead. The advanced platform performs within 3% of the optimal bare-metal result consuming only 5% more energy. The energy efficiency of this advanced platform is quite impressive, considering that a virtualized system must maintain its host's software hypervisor as well as many hardware devices such as the I/O memory management unit, shadow translation lookaside buffer, and the CPUs virtualization extensions.

To confirm the performance implications with newer and more advanced OpenGL implementations, we next run the Unigine Sanctuary Benchmark at 1920 $\times$ 1080, with all high-detail settings enabled. To further stress the GPUs, we enable 8$\times$ AA and 16$\times$ AF. We run the benchmark mode three times and measure the average frame rates and the energy consumption of the system. In this experiment, the running time is consistent between each run and platform; we express the results in watts (joules per second). The results are given in Fig. 2. Once again, we see that our earlier virtualized system shows significant signs of performance degradation compared with its bare-metal optimal. The earlier system drops from 84.4 to 51 frames/s when virtualized, i.e., nearly 40%. The advanced system has a nearly identical performance when the game engine is running in a virtualized environment or directly on the hardware. Table I further shows that our advanced system uses a measurable more power, but still impressively remains at an increase of only 4.4% more power when virtualized. The power consumption of the earlier system appears to have only increased by 3.2%; on the surface, this looks like a positive trait, but indeed comes with the cost of nearly 40% lower video frame rate.

E. GPU Memory Bandwidth

Memory transfer from the system's main memory to the GPU can be a bottleneck for gaming, which can be even more severe when the transfer is performed in a virtualized environment [33]. This would affect the game's performance as well, because both Doom 3 and the Unigine engine must constantly move data from the main memory to the GPU for processing. To understand the impact, we ran a simple memory bandwidth experiment written in OpenCL by NVIDIA.[2] We tested three different copy operations, from host's main memory (DDR3) to the GPU device's global memory (GDDR5), from the device's global memory to the host's main memory, and finally from the devices global memory to another location on the devices' global memory. We give the results for host-to-device and device-to-host experiments in Fig. 3.

For both systems, the bare-metal outperforms the VMs in terms of memory bandwidth across the PCI-E bus to the GPU. The earlier system loses over 40% of its memory transfer

[2]The code is available in the NVIDIA Open Computing Language or OpenCL software development kit at http://developer.NVIDIA.com/opencl

to and from the host memory compared with the bare-metal system. The newer more advanced platform degrades even more, losing over 60% of its memory transfer speed when virtualized. Interestingly, all virtualized systems sustain a nearly identical performance to their bare-metal counterpart for a device-to-device copy: the earlier platform at 66 400 MB/s, while the more advanced platform at 194 800 MB/s. This indicates that once the data is transferred from the virtual machine to the GPU, the commands that operate exclusively on the GPU are much less susceptible to the overhead incurred by the virtualization system.

Our results indicate that even though the gaming performance issues have largely disappeared in the more modern advanced platform, the memory transfer from main memory across the PCI-E bus remains a severe bottleneck. It can become an issue during the gaming industry's transition from 1080p to the newer high memory requirements of 4K (4096 × 2160) ultrahigh-definition (UHD) resolution.

F. Advanced Device Pass-Through Experiments

The previous experiments have shown that running gaming applications inside a virtual machine is now feasible with little overhead. The observation, however, is confined to virtual machines having access to the entire system. In a real-world deployment over a public cloud, we are more interested in sharing a physical machine and its resources among multiple users. To this end, we upgraded our advanced platform to three R9 280× GPUs, so as to test the performance and energy implication of multiple gaming applications running at one time on our platform. We required three GPUs, as we tested the performance of direct device pass through, which is a one-to-one mapping between GPU and VM. To test the performance of other widely deployed 3-D rendering languages, such as Microsoft's DirectX, we further created a Windows server 2008 virtual machine and corresponding bare-metal installation on our platform.

1) 3DMark Windows and DirectX 11 Performance: Our first advanced experiment utilizes Futuremark's widely deployed gaming benchmark `3DMark`.[3] Specifically, we use the `Fire Strike` module, which is a 1080p benchmark utilizing the latest DirectX features (DX11) that stresses not only the 3-D rendering capabilities of the system but also the physics calculations done on the CPU. We run the entire Fire Strike benchmark three times and provide the average graphics and physics frame rates. Once again, we collect the total energy consumption experienced by the system running the benchmark. We first run the benchmark on our bare-metal windows server to determine the optimal baseline. We then perform experiments using virtual machines, assigning each VM a dedicated R9 280× GPU and 4 GB of RAM. We test the performance of different numbers of VMs ranging from one to three concurrently running the 3DMark benchmark.

The performance results for the 3DMark experiment are given in Fig. 4. We first look at the average frame rate; our bare-metal system achieves 34.71 frames/s and our single VM experiment achieves 34.69 frames/s. They are within 0.1% of each other after three runs, that is, they are statistically identical. The two-VM case drops to an average of 34.55 frames/s, which is still only a small drop of about 0.4%. Finally, with three VMs performing GPU-intensive operations, the average score drops to 34.54 frames/s, which is still less than 0.5% drop in performance. In terms of graphic-intensive gaming applications, very little performance is lost when adding more concurrent VMs. Also presented in Fig. 4 is the 3DMark's CPU intensive physics gaming benchmark. Immediately, we see a large difference between the bare-metal and the single virtual machine performance, even though they have access to the same amount of CPU resources. Our virtual system suffers a 30% performance degradation from its optimal bare-metal performance. We have found that this is because the virtualized system does not properly use the hyperthreading provided by the Intel CPU. To show this, we disable the hyperthreading in our system's BIOS and rerun Fire-Strike on the bare-metal platform, which results in a physics test frame rate drop from 32.35 to 23.04 frames/s, making the performance of the bare-metal and the virtual machine within 1% of each other.

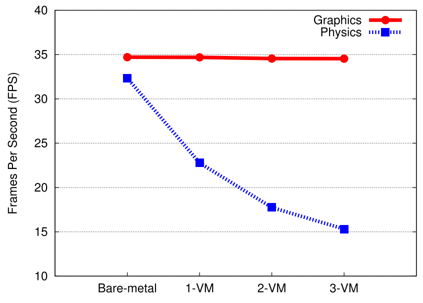


Fig. 4. Advanced: 3DMark frame rate.

Hyperthreading allows multiple software threads to be loaded onto a physical core at one time. Thus, hyperthreading can potentially improve the performance of multithreaded applications. However, the performance gain is application dependent; if functioned fully, it can offer improved CPU utilization. Although in many cases game performance is bounded by the GPU, those games that introduce highly intensive CPU workloads and are properly optimized for hyperthreading may still benefit, and the 3DMark's physics benchmark in our experiment is clearly such a case. Our results suggest that certain gaming related optimizations could be applied to the Xen hypervisor to allow it to fully utilize hyperthreading for gaming applications.

Next we run two VMs concurrently, and we find that the physics performance drops to just over 55% of the bare-metal baseline, and running three VMs drops to under 50% of the bare-metal baseline. Although the single VM experiment is surprising, the performance loss in the two- and three-VM test are more expected. Unlike the GPU intensive graphics modules in this benchmark, the CPU intensive physics modules are all processed by a single physical component, the CPU. Due to this device sharing, multiple VMs competing for the same CPU would affect each others' performance. The 3DMark's physics benchmark has been specifically programmed to use all the available CPUs. Other than the frame rate, this

[3]http://www.3dmark.com/

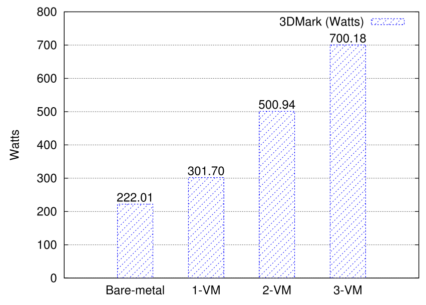


Fig. 5. Advanced: 3DMark power usage.

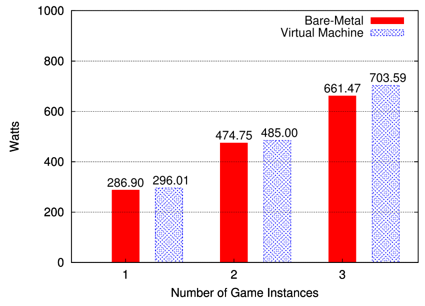


Fig. 6. Advanced: Heaven power usage.

TABLE II
ADVANCED: MULTIPLE INSTANCES AVERAGE FRAME RATE

# of Instances	Bare-metal (fps)	VM (fps)
1	29.9	29.2
2	30.3	28.9
3	29.8	28.9

benchmark also gives the score as a form of general evaluation, and fortunately, a large public accessible dataset is available,[4] which can be used to determine how a particular system performs compared with thousands of others. When searching this database, we find that, under the three VM case, with a score of approximately 5000, each machine is achieving the gaming performance of a midrange Intel Core i5-2300 processor. For many gaming applications, however, the GPU is the bottleneck, and thus a shared CPU may not impact the game performance.

The power consumption for this test is given in Fig. 5, which shows that our bare-metal system uses approximately 35% less energy than the same application running inside the VM. Further, the power consumption increases greatly as we increase the number of concurrently running VMs. This is because each additional VM saturates another dedicated physical GPU. Unfortunately, 3DMark only allows our bare-metal platform to run one instance of the Fire-Strike benchmark at a time, and hence, we could not directly compare multiple instances of the same gaming application directly running on the hardware to those running inside different VMs.

2) Unigine Heaven—Multiple Instance Performance: To determine such a difference in performance and energy consumption, our final experiment leverages the Unigine's latest benchmark Heaven,[5] which supports the latest graphics rendering techniques such as DirectX 11 and OpenGL 4.0. Once again, we use a 1920 $\times$ 1080 resolution, with all high detail settings enabled and also utilize 8$\times$ AA and 16$\times$ AF.

The results in Table II suggest that the performance remains consistent as we add more instances of the application to either the bare-metal or the virtualized system. For the bare-metal system, the average performance is approximately 30 regardless of the number of running game instances. The virtualized performance is approximately 5% lower at around 29 frames/s. Fig. 6 further gives a comparison of the energy consumption. Although the energy consumption is higher in the virtual machine deployment, at its worst, it is only about 6.5% more. This is impressive considering that in the virtual machine deployment, we actually have one operating system for the virtual machine host and one for each guest VM. Overall, given such important benefits of virtualization as performance isolation and resource sharing, the overhead of around 5%–6.5% for advanced virtualization is quite justifiable.

Our measurements above have demonstrated the strong potentials of today's GPU for high-quality cloud gaming. Following up on these results, in the next section, we are able to deploy a cloud gaming platform using public cloud resources, namely, the Amazon G2 Instance. These instances are powered by a NVIDIA GRID class GPU. This switch in hardware is associated with using a public cloud, where we do not control the architecture. However, we continued to use the same nonproprietary GPU pass-through technology.

IV. VIRTUALIZED CLOUD GAMING PLATFORM: DESIGN AND IMPLEMENTATION

Our measurements above have demonstrated the strong potentials of today's GPU for high-quality cloud gaming. Migration to a real-world public cloud with resource virtualization, however, is nontrivial considering the interaction and the integration of the many modules in a gaming system. To establish the best practices of system level tradeoffs of cloud gaming over real-world public and virtualized cloud platforms, we have implemented a high-performance cloud gaming system named Rhizome. Our implementation is a fully functional and modular platform, which allows users and researchers to customize many subsystems. It not only supports software encoding using the highly optimized `x264` encoder for H.264 video[6] but also the state-of-the-art NVIDIA GRID and its hardware H.264 encoder.[7] The streaming protocol can also be customized, e.g., Real Time Streaming Protocol over User Datagram Protocol (UDP) or Transmission Control Protocol (TCP), as well as over Hypertext Transfer Protocol (HTTP)/TCP streaming, using the widely deployed open-source streaming library `Live555`[8] as the streaming engine.

We now offer a high-level description of our Rhizome system, whose design is inspired by existing cloud gaming

[4]http://www.3dmark.com/search
[5]https://unigine.com/products/heaven/
[6]http://www.videolan.org/developers/x264.html
[7]http://www.nvidia.ca/object/grid-processors-cloudgames.html
[8]http://www.live555.com/

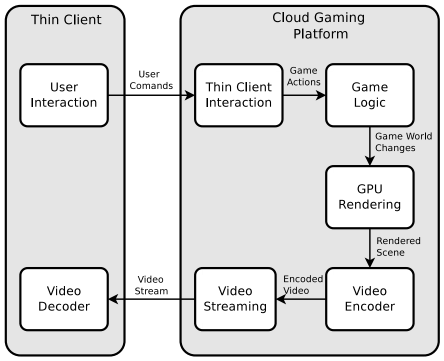


Fig. 7. Rhizome architecture.

systems, yet incorporates the latest advances of virtualization and hardware. Rhizome allows users and researchers to customize its many subsystems as well. A sketch of the Rhizome's architecture is given in Fig. 7. Our findings facilitate the extension of other open-source cloud gaming systems (e.g., *GamingAnywhere*) to the public cloud as well.

A. Remote Server Setup

We start with the implementation of the *user interactions* module, which targets both mobile (Android) and PC users. It captures the user commands using `VNC` remote desktop hooks, which have inherent cross-platform support. The commands are sent back to the server over TCP, where the server interprets the commands and sends them to the gaming window. We emphasize here that `VNC` is only used for sending commands to the server, for its low overhead and latency; it is not involved in transmission/reception of the video any way, as its performance is not enough for such data-intensive jobs.

The next module is the *game logic*, which in essence is the gaming application that the user is playing. It intercepts the user's keystrokes to the game and computes the game world changes. The rendering is performed by the GPU that is assigned to the VM, e.g., an NVIDIA GRID GK104. The video encoder, which in our platform is selectable, consisting of either a software or a hardware H.264 encoder. The software encoder is provided by the high-performance `x264` encoding library; if the VM is assigned a GRID GPU, it can utilize the built-in H.264 encoder provided by the platform. In either case, the encoders need additional support for real-time streaming, namely, a *discreet framer* module, which allows the Live555 streaming library to request live frames from the encoders at a desired video stream frame rate. The encoded video stream is then encapsulated and transported in either UDP, TCP, or HTTP/TCP. Finally, when the video is received by a thin client (mobile or PC), we use the cross-platform media library `FFmpeg`[9] with real-time optimizations to decode the video and display it on the client device.

Our design and implementation are platform independent, although a GRID GPU is required for hardware encoding. We have deployed and experimented with the system on Amazon EC2 GPU Instances (G2), a new type of cloud instances backed by the Intel Xeon E5-2670 (Sandy Bridge) processors and the NVIDIA GRID-K520 board that contains a GK104 GPU with 1536 CUDA cores and 4 GB of video memory.[10] The GRID's on-board hardware video encoder supports up to eight live HD video streams (720p at 30 frames/s) or up to four live Full HD (FHD) video streams (1080p at 30 frames/s), as well as low-latency frame capture for either the entire screen or selected rendering objects, enabling a G2 instance to offer such high-quality interactive streaming as game streaming, 3-D application streaming, or other server-side graphics tasks.

We emphasize here that the use of virtualization opens a new space for the design and deployment of cloud gaming. It can now be readily deployed or migrated to a public cloud (e.g., Amazon EC2) with low cost and practically unlimited resources. There is no need to maintain a dedicated and specialized private cloud. Although the GRID GPU is passed in a physical device, other parts of the system can easily be tuned by the cloud provider. For instance, if future 3-D games require more main memory, this can easily be satisfied as the amount of memory is a tunable parameter for a cloud instance. Similarly, if future games require more computing resources, the cloud provider can increase the VM's number of virtual CPUs or their share of the physical resources.

V. Hardware Versus Software Video Encoding

The video encoder is the key component in the cloud game engine and should be the focus of optimization. Existing video encoding applications have relied heavily on software-based encoders, e.g., the `x264` encoder, which is also used in *GamingAnywhere* [22]. Although OnLive has used a hardware encoder, it is not built into the GPU, but instead attached to the physical output of the video card. Our system supports both hardware and software encoding, which allows us to directly compare the performance of hardware and software H.264 encoders in terms of both encoding latency and CPU overhead.

A. Encoder Benchmark

We configure our Rhizome platform to capture video from the GPU and encode it using either the hardware or software H.264 encoder. For all the tests, we again used the Unigine Heaven benchmark playing in a loop as the rendering source and used the CPU's hardware counters to accurately determine the running time of encoding a frame. We also collected the CPU usage during the test, for both the encoding task alone and the complete usage of the encoder as well as of the streaming application.

For the software `x264` encoder, we performed measurements for a multithreaded implementation. It is worth mentioning that although a single-threaded implementation is not likely to cause large performance interference with gaming applications, more threads are required to reduce frame encoding latency to acceptable levels for cloud gaming.

For both the software encoder and the hardware encoder, we recorded videos at 30 frames/s at a variable bitrate (VBR)

[9]https://www.ffmpeg.org/

[10]http://aws.amazon.com/ec2/instance-types/

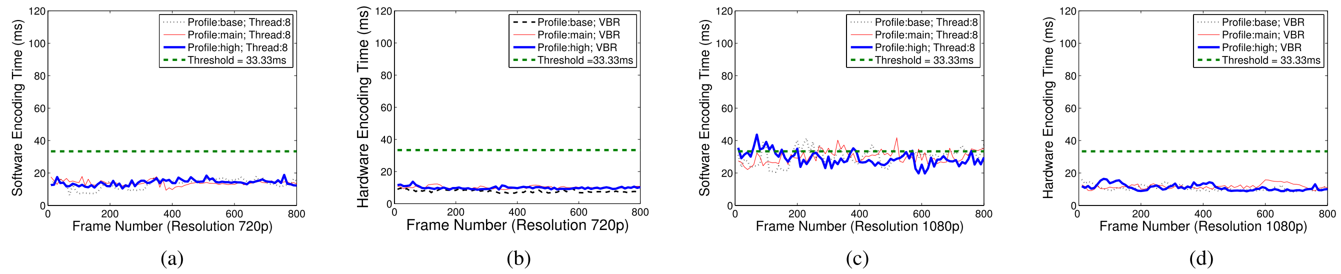


Fig. 8. Encoder comparison. (a) Encoder: software 720p. (b) Encoder: hardware 720p. (c) Encoder: software 1080p. (d) Encoder: hardware 1080p.

TABLE III
CPU OVERHEAD: HARDWARE VERSUS SOFTWARE H.264

Encoder Type	Encoder (CPU %)	Encoder + Streaming (CPU %)
Software(720p)	˜15%	˜23%
Hardware(720p)	˜1%	˜10%
Software(1080p)	˜25%	˜37%
Hardware(1080p)	˜1%	˜14%

of 8 Mbits/s, and under three H.264 encoding profiles, namely, *base*, *main*, and *high*. The distinction is important because different platforms may implement only certain profiles, e.g., many mobile devices only support the base profile, as it is the simplest and most energy efficient. We recorded the running time for each call to the encoder and plotted them against each other. Further, we set a threshold value of 33.33 ms, which is the maximum single frame encoding time to allow 30-frame/s video. Any time higher than 33.33 ms indicates that the current encoder cannot produce video at 30 frames/s. The parameters of the `x264` software encoder were initialized with the `zerolatency` and `very-fast` flags for the fastest possible encoding speed. For the GRID's built-in hardware H.264 encoder, we applied NVIDIA's default encoding parameters, which provide a good balance between encoding speed and image quality.

1) 720p Result: We start our discussion with 720p H.264 software encoding; the multithreaded experimental results can be found in Fig. 8(a). Regardless of the profile chosen, the encoder manages an average frame encoding time between 13–14 ms. The longest encoding time experienced by any frame in this test is approximately 20 ms, making even the worst case acceptable for smooth 30-frames/s encoding.

Next, we tested the GRID's built-in hardware H.264 encoder and the results are given in Fig. 8(b). As can be seen, the results are very consistent regardless of the encoding profile, with an average encoding time less than 10 ms. Overall, the hardware encoder, on average, encodes frames 30% faster than the software encoder does, with a lower deviation.

Table III further lists the corresponding CPU utilization, which provides the CPU times taken by both the encoder alone and the complete system (encoder + streaming overhead). As can be seen, the software encoder takes considerably more CPU resources than the hardware encoder. In fact, 15% of the total CPU is consumed by the multithreaded encoder, which implies that more than one entire core of the eight-cores is being used by the video encoding subsystem. It is also important to note that even though both hardware and multithreaded software encoders on our cloud platform are capable of encoding 720p HD video at 30 frames/s, for cloud gaming, the lower the latency at this stage, the lower the playback delay will be at the end user. This is because after the streaming software requests a new frame, it has to wait for whatever amount of time it takes to encode before sending to the end user.

2) 1080p Result: We now look at the more complex situation of encoding FHD 1080p video, using the same settings as the previous experiment, except for the resolution.

Once again, we start our discussion with the multithreaded software encoder. The results can be seen in Fig. 8(c). For the high profile encoding, the average frame encode time sits over 29 ms, and the main and base profile both attain an average of 30 ms/frame. The multithreaded software encoder's performance at 1080p manages to stay under 33.33 ms/frame on average for all the profiles. However, all profiles have encoding time spikes that are over 40 ms. This means the software encoder will not always meet the encoding deadline to provide fluid 30-frame/s video.

The results for the hardware H.264 encoder can be found in Fig. 8(d). Regardless of encoding profile level, the average encoding time is less than 11 ms/frame. During our experiments, the worst case encoding time for the hardware encoder is only slightly over 25 ms. It is clear from these results that GRID has no issue with encoding full high-definition frames at rates above 30 frames/s.

A closer look into the results is shown in Table III, which reveals the CPU processing overhead of both the software-based `x264` encoder and the hardware encoder while processing 1080p video frames. For the multithreaded `x264` encoder, it consumes 25% of the available CPU resources for the encoding phase and a combined CPU utilization of 37%. This implies that on the eight-core system, the software encoder consumes the equivalent resources of two cores to provide the encoding and nearly three cores to provide both the streaming and encoding systems. Despite this high CPU usage, the software-based encoder often misses the 33.33-ms encoding deadline for 30-frame/s video. On the other hand, the hardware H.264 encoder easily handles the frames with a very low CPU utilization and never exceeds the threshold.

In summary, being able to offload the computationally expensive job of encoding frames to a dedicated piece of hardware not only shows a great reduction in CPU consumption but also completes encoding a frame with much lower latency. This reduction in latency will result in improved interaction

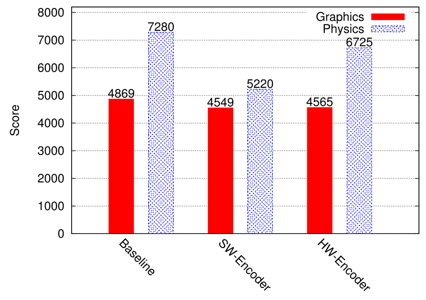


Fig. 9. Encoder choice performance implication.

delay for the end user, as the streaming subsystem is not stalled while waiting for the next frame. Further, the `x264` software encoder's performance, even when using eight threads, does not allow smooth capture of a 1080p video, whereas the hardware encoder handles these high-definition frames with low latency. As the resolution increases (e.g., toward 4K UHD video), the suitability of the software video encoder will quickly diminish.

B. Impact of Encoder Choice on Gaming Performance

To determine the real-world performance implications that different encoding methods have on cloud gaming, we again utilize Futuremark's widely used gaming benchmark `3DMark`. For a comparison, we encode a 1080p stream at 30 frames/s using a VBR of 8.0 Mbits/s on the H.264 main preset for both our hardware and software encoder. We also compute a baseline performance of 3DMark running on our platform with all streaming and encoding tasks being disabled. Although disabling all video outputs from our platform implies that one would have to physically be in the data-center with a monitor to see the game, it does provide a useful best case scenario for what the maximum gaming performance of the EC2 instance is. Once again, we report the score of both the graphic and physics experiments contained in the 3Dmark benchmark.

The results are given in Fig. 9. In terms of graphics performance, we see that both the hardware and software encoders fall by a very similar amount, approximately 6%. We conjecture the loss of graphics performance between these two is similar because both must download the current frame from the GPU. In the case of the hardware encoder, the frame is a fully compressed H.264 frame, but in the software encoding case, the frame is a raw YUV image. In either case, this download causes small but notable interference with the GPU's main task of rendering 3-D images for the 3Dmark benchmark. The impact on game physics caused by the choice of encoding method is much more profound. For example, when the Rhizome platform employs a hardware encoder, the drop in physics performance is just under 8%; on the other hand, if we employ the software encoder, the loss is even greater, falling by over 28%. This major loss in physics performance is due to the game engine contending for CPU time with the software encoder.

As such, not only do hardware-encoding-based cloud gaming platforms greatly reduce encoding latency, but also they help preserve gaming performance. This fact is especially important for gaming applications that are performance constrained by complex physics operations.

VI. Thin Client: Configuration and Performance

Finally, we examine the configuration and performance of client terminals in our system. Migrating the game engine to the cloud enables thin clients to play advanced games. However, they still need to perform decoding in real time, which remains a nontrivial task for their given highly constrained resources in terms of computation, memory, battery, and so on. As we will show, without careful planning, the overhead of decoding could overshadow the benefit of computation offloading to the cloud.

Our test system was a 7-in EVGA Tegra NOTE 7 tablet, powered by NVIDIA's advanced mobile chip-set Tegra 4, functionally a system on chip that features a quad-core ARM Cortex-A15 CPU, a 72-core NVIDIA GeForce GPU, and, particularly, a dedicated video decoding block that provides full support for hardware decoding. And to determine the overall system's power consumption in executing client tasks, we resorted to a direct at the circuit level means: measuring the direct current (dc) consumption of the tablet's circuit. Given the fact that the tablet's working dc voltage stays mostly stable (=3.7 V) for a sufficiently long period when the battery's remaining capacity is over 50%, we can compute the instantaneous power as the product of current (in amperage) and voltage. To this end, we crafted the tablet to use a digital clamp meter (Mastech MS2115B) to measure the dc amperage (precision of $\pm$2.5%) external display through its video output. The installed Android 4.2.2 Jelly Bean enables users to explore and utilize the device with a wide range of gaming applications. All these make it an ideal cloud gaming client for both use and test.

A. Gaming Applications and Benchmark Setup

We configured our Tegra tablet to make use of its built-in hardware decoder and to switch between hardware decoding and software decoding. For comparing their respective energy consumptions, we chose a high-definition video clip as our sample video, which is original in 1080p resolution. To have a consistent comparison across high and low resolutions, we used the VLC media player to convert this source video to multiple versions, including 480p H.264 + advanced audio coding (AAC) (MP4) at 3.0-Mbit/s bitrate, 720p H.264 + AAC (MP4) at 6.0-Mbit/s bitrate, and 1080p H.264 + AAC (MP4) at 10.0-Mbit/s bitrate.

We also used an advanced 3-D benchmark for mainstream tablets, namely, `3DMark Ice Storm`, which includes 720p game scenes, two graphics tests, and a physics test to stress GPU and CPU. We installed it both on the Tegra tablet for the power measurement of local rendering and on the Rhizome server for remote rendering.

B. Client-Side Power Consumption

For each run of the experiments, we fully recharged the tablet's battery so as to ensure a stable voltage supplied. Then we restarted the tablet to minimize the interference from running other background applications and unplugged the power supply to ensure the power source purely being the battery. For the ice storm experiment, when we did the local rendering tests, we started only the 3DMark application, ran the benchmark, and began to record readings off the clamp meter using the data logger PCLink with a sampling frequency of 500 ms. Likewise, when conducting the remote rendering tests, we selected the network video stream and started measurement recording. By turning on and off the hardware acceleration option before each run, we obtained the measurements with the hardware decoder or with the software decoder. For all experiments, the screen brightness, sound volume, and other settings remained unchanged.

For the experiment of high-definition video clips, we followed the same steps to run each version of the sample clip, except that it did not involve any local rendering tests. It is worth mentioning that the dc clamp meter is highly sensitive; hence, to reduce the measuring error, we reset the clamp meter before each measurement.

1) Video Clip Results: In Table IV, we show the resulting CPU usage (percentage) and power consumption (wattage) of locally playing the video clips, using software decoder and hardware decoder, respectively, on different resolutions. The power consumption started from 2.81 W at 480p and increased to 4.19 W at 720p, then mounted to 6.21 W at 1080p. The hardware decoding had a lower start of 1.87 W at 480p, 2.19 W at 720p, and ended with a slightly higher 2.35 W at 1080p. Clearly, the hardware decoder outperforms its software counterpart, by consuming approximately half the amount of power. This can be explained by the fact that the software decoding invoked much more intensive CPU workloads than the hardware decoding, as the CPU usage, obtained from the Android console, indicates the hardware decoder slightly incurred more system calls but much fewer user calls than the software decoder; more importantly, the total CPU overhead remained nearly constant, regardless of how high the resolution was. On the contrary, for the software decoder, the total CPU overhead surged from 66% at 480p to 89% at 1080p.

From an even more practical perspective, given the 15.1 Wh battery capacity, when a user successively play the 480p videos using the software decoder, our tablet can only work for roughly 5.4 h (15.1 Wh divided by 2.81 W). With the hardware decoder, it can be prolonged to roughly 8 h. Likewise, for the 720p video, it is 3.6 h against 7 h and 2.4 h against 6.4 h (more than double) for the 1080p display. This margin significantly grows when the video's resolution increases; for a 4K ultrahigh resolution display, even with a short play duration, the hardware decoding will make a quantitative difference against the software decoding with regard to devices' battery life.

2) Ice Storm Results: In Fig. 10, we show the results of the advanced 3-D benchmark for mainstream tablets, namely, the 3DMark Ice Storm. The HD H.264 stream was created using our Rhizome platform introduced in the previous section. The horizontal axis represents the timeline of the benchmark in seconds and the vertical denotes the overall power consumption in watts. The hardware decoding has the lowest power

TABLE IV
POWER CONSUMPTION AND CPU OVERHEAD: HARDWARE VERSUS SOFTWARE H.264

Decoder Type	Avg. Power (Watts)	CPU Usage (Percentage)
Software(480p)	2.81	˜66%
Hardware(480p)	1.87	˜60%
Software(720p)	4.19	˜73%
Hardware(720p)	2.19	˜61%
Software(1080p)	6.21	˜89%
Hardware(1080p)	2.35	˜59%

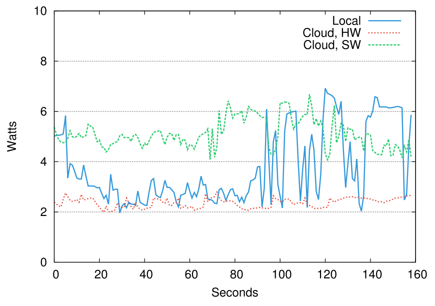


Fig. 10. Ice storm power consumption.

consumption (average 2.36 W), which stayed stable throughout the entire benchmarking. The local rendering is represented as the solid line sitting roughly in the middle (average 3.75 W), which started high at around 5 W, dropped under four, and remained there until the later part of the benchmark. The other line sitting roughly on the top (average 5.11 W) gives the result with software decoding. When applying the hardware decoding, the device had an overall power consumption between 2 and 3 W. Local rendering stayed competitive with the hardware decoding at the midpoint of the benchmark, where only game logic scenes were rendered, and no benchmark loading or other ultra-CPU-intensive work was taking place. However, the software decoding doubled the amount of power consumption to over 5 W, which was even worse than the local rendering in terms of supporting average game logic scenes.

C. Hardware Decoder: Key for Cloud Gaming Client

A deeper investigation shows that local rendering increases the power consumption at the very beginning and particularly in the later part of the benchmark. We believe that this is due to a great amount of CPU and GPU calls being made to load and initialize the benchmark, stressing the device in its graphics and physics test. This conjecture is not only based on our track of CPU/GPU usage but also on the fact that the sharp increase of power consumption corresponds to the time that the benchmark spends loading and stress testing, where extraordinarily intensive workloads were involved. In other words, as opposed to traditional locally rendered gaming, cloud gaming saves a large amount of power consumed not only in rendering game scenes but also in game loading and other CPU/GPU intensive operations.

However, the savings come from hardware decoding only. The software decoder on the client side consumes even more energy than local rendering by 5.11 W against 3.75, theoretically reducing the battery life by 27% and making cloud gaming less appealing for the given notorious shortage of power supply in mobile devices. The hardware decoder, on the other hand, obtains a much lower average power consumption at 2.36 W. This impressively extends the battery life by 59% longer than local rendering and 117% longer than using the software decoder, which makes battery shortage much less of a concern for users.

Moreover, as exhibited in the video clip experiment, when video resolution increased from 480p to 720p and to 1080p, the software decoder expended significantly more computing overhead and power consumption, while the hardware decoder managed to contain the computational and energy cost to a relative low amount. Given that mainstream tablets now utilize 720p resolution and are moving toward 1080p and 4K, the gap between software and hardware decoders will only increase. For richer and more detailed game world at 4K UHD resolution and beyond, a hardware decoder is definitely needed, though it remains to be universally supported by tablets and smart phones.

VII. Conclusion

Cloud gaming has attracted significant interest from both academia and industry, and its potentials and challenges have yet to be fully realized, particularly with the latest hardware and virtualization technology advances. This paper presented an in-depth study in this direction. We closely examined the performance of modern virtualization systems equipped with virtualized GPU and pass-through techniques. Our results showed that virtualization for GPU has greatly improved and is ready for gaming over a public cloud. A modern platform can host three concurrent gaming sessions running inside different VMs at 95% of the optimal performance of a bare-metal counterpart. Although there is degradation of memory transfer between a virtualized system's main memory and its assigned GPU, the game performance at the full HD resolution of 1080p was only marginally impacted.

Based on our findings, we have designed and implemented a practical cloud gaming system, Rhizome, whose server side was deployed with virtualization and advanced GPU support in a public cloud platform. Our real-world experimental results revealed the clear advantages of the hardware encoding over its software counterpart. We showed that only hardware encoders can achieve acceptable gaming performance with 1080p at 30 frames/s. We also explored practical issues facing the thin client design. We measured and quantified the power consumption of different ways to supply game scenes, and found that a software decoder can consume even greater power than local rendering. Only with a hardware decoder will a thin client benefit from the power saving of remote rendering, particularly for high definitions.

Our findings and implementation can be generalized to other open-source cloud gaming systems, such as *GamingAnywhere*. For a future work, we plan on modifying other platforms to support advanced hardware encoders and analyze their performance. Recent research has also shown that decoding complexity and battery life of mobile devices can be greatly effected by encoding parameters set at the server side [34]. It is likely that further energy savings can be made with intelligent tweaking of the encoders settings.

It is noteworthy that cloud gaming, compared with local gaming platforms, still falls behind in providing equivalent gaming experience for high-end users. A delay of $\approx$50 ms, which already can be considered very low for cloud gaming, is still detectable [35] and is treated as an intolerable game lag for certain high-demanding users. Discovering novel transmission and intelligent masking techniques that physically or perceptually reduce the interaction delay thus remains a key direction to explore. For instance, by striping data across a set of simultaneous connections, parallel TCP increases the end-to-end throughput [36] and is likely to be beneficial for data-intensive cloud gaming applications in terms of throughput and delay. Recent advances in TCP tail loss recovery mechanisms have demonstrated further latency reduction for TCP against traffic bursts [37], and several Linux distributions have readily provided advanced TCP streaming extensions, such as thin dupACK and thin retransmit.

Cloud gaming is rapidly evolving, particularly toward 4K UHD resolution that will offer highly immersive gaming experience. Despite that 4K display devices are readily available in the market, processing and delivering games in real time impose unprecedent pressure to each and every module of existing cloud gaming platforms. We have begun working on a 4K-enabled cloud gaming server and are currently undertaking research to discover if any virtualization specific issues exist at these ultrahigh resolutions. It is likely that practical changes can be made at the hypervisor level to enhance cloud gaming performance. Such enhancements may include game-aware virtual CPU scheduling as well as memory management techniques to increase memory transfer speed from the VM to the GPU.

References

[1] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] W. Cai, M. Chen, and V. C. M. Leung, "Toward gaming as a service," *IEEE Internet Comput.*, vol. 18, no. 3, pp. 12–18, May/Jun. 2014.

[3] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, Jul./Aug. 2013.

[4] M. Claypool, D. Finkel, A. Grant, and M. Solano, "On the performance of OnLive thin client games," *Multimedia Syst.*, vol. 20, no. 5, pp. 471–484, 2014.

[5] K. Lee, D. Chu, E. Cuervo, A. Wolman, and J. Flinn, "Demo: DeLorean: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 12th Annu. Int. Conf. ACM MobiSys*, 2014, p. 347.

[6] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere: An open cloud gaming system," in *Proc. 4th ACM Multimedia Syst. Conf. (MMSys)*, 2013, pp. 36–47. [Online]. Available: http://doi.acm.org/10.1145/2483977.2483981

[7] OnLive. [Online]. Available: http://www.onlive.com/, accessed Sep. 2014.

[8] Gaikai. [Online]. Available: http://www.gaikai.com/, accessed Sep. 2014.

[9] Engadget. *Sony Buys Gaikai Cloud Gaming Service for $380 Million*. http://www.engadget.com/2012/07/02/sony-buys-gaikai/, accessed Sep. 2014.

[10] M. Dowty and J. Sugerman, "GPU virtualization on VMware's hosted I/O architecture," *ACM SIGOPS Oper. Syst. Rev.*, vol. 43, no. 3, pp. 73–82, 2009.

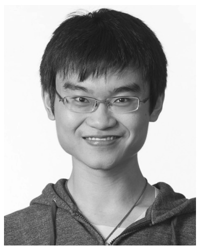
[11] D. De Winter *et al.*, "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications," in *Proc. Int. Workshop NOSSDAV*, 2006, Art. ID 15.

[12] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, and C.-L. Lei, "Understanding the performance of thin-client gaming," in *Proc. IEEE Int. Workshop Tech. Committee CQR*, May 2011, pp. 1–6.
[13] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei, "Are all games equally cloud-gaming-friendly? An electromyographic approach," in *Proc. 11th Annu. Workshop NetGames*, 2012, pp. 1–6.
[14] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "Gaming in the clouds: QoE and the users' perspective," *Math. Comput. Model.*, vol. 57, nos. 11–12, pp. 2883–2894, 2013.
[15] M. Hemmati, A. Javadtalab, A. A. N. Shirehjini, S. Shirmohammadi, and T. Arici, "Game as video: Bit rate reduction through adaptive object encoding," in *Proc. 23rd ACM NOSSDAV*, 2013, pp. 7–12.
[16] D. Wu, Z. Xue, and J. He, "iCloudAccess: Cost-effective streaming of video games from the cloud with low latency," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1405–1416, Aug. 2014.
[17] 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 Trans. Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.
[18] B. Vankeirsbilck *et al.*, "Platform for real-time subjective assessment of interactive multimedia applications," *Multimedia Tools Appl.*, vol. 72, no. 1, pp. 749–775, 2014.
[19] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proc. 11th Annu. Workshop Netw. Syst. Support Games*, 2012, pp. 1–6.
[20] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan. 2015.
[21] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia Syst.*, vol. 20, no. 5, pp. 503–519, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00530-014-0367-z
[22] C.-Y. Huang, C.-H. Hsu, D.-Y. Chen, and K.-T. Chen, "Quantifying user satisfaction in mobile cloud games," in *Proc. ACM Workshop MoViD*, 2014, Art. ID 4.
[23] M. Manzano, M. Urueña, M. Sužnjević, E. Calle, J. A. Hernández, and M. Matijasevic, "Dissecting the protocol and network traffic of the OnLive cloud gaming platform," *Multimedia Syst.*, vol. 20, no. 5, pp. 451–470, 2014.
[24] M. Dusi, S. Napolitano, S. Niccolini, and S. Longo, "A closer look at thin-client connections: Statistical application identification for QoE detection," *IEEE Commun. Mag.*, vol. 50, no. 11, pp. 195–202, Nov. 2012.
[25] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in Xen-based virtual cluster environment," in *Proc. 12th IEEE Int. Conf. High Perform. Comput. Commun.*, Sep. 2010, pp. 273–280.
[26] P. Barham *et al.*, "Xen and the art of virtualization," in *Proc. 19th ACM Symp. Oper. Syst. Principles*, 2003, pp. 164–177.
[27] *Amazon Elastic Compute Cloud*. [Online]. Available: http://aws.amazon.com/ec2/, accessed Sep. 2014.
[28] L. Shi, H. Chen, J. Sun, and K. Li, "vCUDA: GPU-accelerated high-performance computing in virtual machines," *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 804–816, Jun. 2012.
[29] C. Reaño, A. J. Peña, F. Silla, J. Duato, R. Mayo, and E. S. Quintana-Ortí, "CU2rCU: Towards the complete rCUDA remote GPU virtualization and sharing solution," in *Proc. 19th Int. Conf. High Perform. Comput. (HiPC)*, 2012, pp. 1–10.
[30] C.-T. Yang, H.-Y. Wang, and Y.-T. Liu, "Using PCI pass-through for GPU virtualization with CUDA," in *Network and Parallel Computing*. Berlin, Germany: Springer-Verlag, 2012, pp. 445–452.
[31] M. Yu, C. Zhang, Z. Qi, J. Yao, Y. Wang, and H. Guan, "VGRIS: Virtualized GPU resource isolation and scheduling in cloud gaming," in *Proc. 22nd Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2013, pp. 203–214.
[32] C. Zhang, Z. Qi, J. Yao, M. Yu, and H. Guan, "vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3036–3045, Nov. 2014.
[33] R. Shea and J. Liu, "On GPU pass-through performance for cloud gaming: Experiments and analysis," in *Proc. 12th Annu. Workshop Netw. Syst. Support Games (NetGames)*, Dec. 2013, pp. 1–6.
[34] M. J. Langroodi, J. Peters, and S. Shirmohammadi, "Complexity aware encoding of the motion compensation process of the H.264/AVC video coding standard," in *Proc. Netw. Oper. Syst. Support Digit. Audio Video Workshop*, 2014, p. 103.
[35] K. Raaen, R. Eg, and C. Griwodz, "Can gamers detect cloud delay?" in *Proc. 13th Annu. Workshop Netw. Syst. Support Games (NetGames)*, Dec. 2014, pp. 1–3.
[36] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Proc. 16th Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2002, p. 314.
[37] M. Rajiullah, P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "An evaluation of tail loss recovery mechanisms for TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 1, pp. 5–11, 2015.

Ryan Shea (S'08) received the B.Sc. degree in computer science from Simon Fraser University, Burnaby, BC, Canada, in 2010 and the certificate in university teaching and learning from Simon Fraser University, where he is currently working toward the Ph.D. degree.

His research interests include computer and network virtualization, and performance issues in cloud computing.

Mr. Shea received the Natural Sciences and Engineering Research Council of Canada Alexander Graham Bell Canada Graduate Scholarship in 2013. He was a recipient of the best student paper award at the IEEE/ACM 21st International Workshop on Quality of Service for his paper Understanding the Impact of Denial of Service Attacks on Virtual Machines in 2012.



Di Fu (S'15) is currently working toward the bachelor's degree in dual degree program with Simon Fraser University, Burnaby, BC, Canada, and Zhejiang University, Hangzhou, China, and the master's degree with Simon Fraser University under the supervision of Prof. J. Liu.

His research interests include networking research, in particular, in cloud computing.

Mr. Fu received several scholarships and awards, including the Elma Krbavac Undergraduate Scholarship in Computing Science in 2014, the Mark and Nancy Brooks Computing Science Innovation Award in 2014, and the Sohpos Annual Computing Science Scholarship in 2013.

Jiangchuan Liu (S'01–M'03–SM'08) received the B.Eng. (*cum laude*) degree in computer science from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, Hong Kong, in 2003.

He was an Assistant Professor with The Chinese University of Hong Kong, Hong Kong, from 2003 to 2004. He was an EMC-Endowed Visiting Chair Professor with Tsinghua University from 2013 to 2016. He is currently a Professor with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, and an NSERC E. W. R. Steacie Memorial Fellow. His research interests include multimedia systems and networks, cloud computing, social networking, online gaming, big data computing, wireless sensor networks, and peer-to-peer and overlay networks.

Dr. Liu was a co-recipient of the Inaugural Test of Time Paper Award of the IEEE INFOCOM (for the INFOCOM paper on CoolStreaming, which has been cited 2000+ times) in 2015, the ACM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, the ACM Multimedia Best Paper Award in 2012, the IEEE Globecom Best Paper Award in 2011, and the IEEE Communications Society Best Paper Award on Multimedia Communications in 2009. His students received the best student paper award of the IEEE/ACM IWQoS twice in 2008 and 2012. He has served on the Editorial Boards of IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE ACCESS, IEEE INTERNET OF THINGS JOURNAL, *Computer Communications*, and *Wiley Wireless Communications and Mobile Computing*. He was the Steering Committee Chair of the IEEE/ACM IWQoS from 2015 to 2017.