Cloud Gaming: Architecture and Performance

Ryan Shea
Simon Fraser University
Burnaby, Canada
Email: rws1@cs.sfu.ca

Jiangchuan Liu
Simon Fraser University
Burnaby, Canada
Email: jcliu@cs.sfu.ca

Edith C.-H. Ngai
Uppsala University
Uppsala, Sweden
Email: edith.ngai@it.uu.se

Yong Cui
Tsinghua University
Beijing, China
Email: cuiyong@tsinghua.edu.cn

***Abstract*—Recent advances in cloud technology have turned 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. This is an advantage for less powerful computational devices that are otherwise incapable of running high quality games. Such industrial pioneers as Onlive and Gaikai have seen success in the market with large user bases. In this article, we conduct a systematic analysis of state-of-the-art cloud gaming platforms, and highlight the uniqueness of their framework design. We also measure their real world performance with different types of games, for both interaction latency and streaming quality, revealing critical challenges toward the widespread deployment of Cloud Gaming.**

I. Introduction

Through the utilization of elastic resources and widely deployed data-centers, *cloud computing* has provided countless new opportunities for both new and existing applications. Existing 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 computing platforms. Much of these advances have come from exploring the cloud's massive resources with *computational offloading* and reducing user access latencies with strategically placed cloud data-centers. Recently, advances in cloud technology have expanded to allow offloading not only of traditional computations but also of such more complex tasks as high definition 3D 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. Figure 1 shows a high level architectural view of such a cloud gaming system with thin clients and cloud-based rendering.

Onlive [1] and Gaikai [2] are two industrial pioneers of cloud gaming, both having seen great success with multi-million user bases. The recent 380 millon dollar purchase of Gaikai by Sony [3], an industrial giant in digital entertainment and consumer electronics, shows that cloud gaming is beginning to move into the mainstream. From the perspective of industry, cloud gaming can bring immense benefits by expanding the user base to the vast number of less-powerful devices that support thin clients only, particularly smartphones and tablets. As an example, the recommended system configuration for *Battlefield 3*, a highly popular first-person shooter game, is a quad-core CPU, 4 GB RAM, 20 GB storage space, and a graphics card with at least 1GB RAM (e.g., NVIDIA GEFORCE GTX 560 or ATI RADEON 6950), which alone costs more than $500. The newest tablets (e.g., Apple's iPad with Retina display and Google's Nexus 10) cannot even meet the minimum system requirements that need a dual-core CPU over 2.4 GHz, 2 GB RAM, and a graphics card with 512 MB RAM, not to mention smartphones of which the hardware is limited by their smaller size and thermal control. Furthermore, mobile terminals have different hardware/software architecture from PCs, e.g., ARM rather than x86 for CPU, lower memory frequency and bandwidth, power limitations, and distinct operating systems. As such, the traditional console game model is not feasible for such devices, which in turn become targets for Gaikai and Onlive. Cloud gaming also reduces customer support costs since the computational hardware is now under the cloud gaming provider's full control, and offers better Digital Rights Management (DRM) since the codes are not directly executed on a customer's local device.

However, cloud gaming remains in its early stage and there remain significant theoretical and practical challenges towards its widespread deployment. In this article, we conduct a systematic analysis of state-of-the-art cloud gaming platforms, both in terms of their design and their performance. We first offer an intuitive description of the unique design considerations and challenges addressed by existing platforms. We highlight their framework design. Using Onlive as a representative, we then measure its real world performance with different types of games, for both interaction latency and streaming quality. Finally, we discuss the future of cloud gaming as well as issues yet to be addressed.

II. Cloud Gaming: Issues and Challenges

From low latency live video streaming to high performance 3D rendering, cloud gaming must bring together a plethora of bleeding edge technologies to function. We begin our analysis with the important design considerations, which are currently being addressed by cloud gaming providers. A cloud gaming system must collect a player's actions, transmit them to the cloud server, process the action, render the results, encode/compress the resulting changes to the game-world, and stream the video (game scenes) back to the player. To

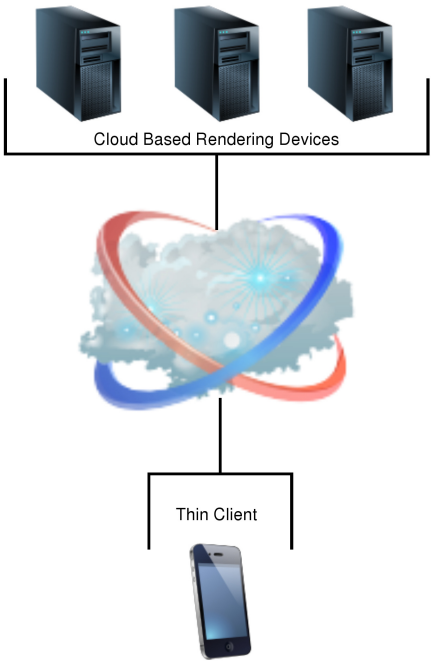


Fig. 1. Cloud Gaming Overview

Example Game Type	Perspective	Delay Threshold
First Person Shooter (FPS)	First Person	100 ms
Role Playing Game (RPG)	Third-Person	500 ms
Real Time Strategy (RTS)	Omnipresent	1000 ms

TABLE I
DELAY TOLERANCE IN TRADITIONAL GAMING

ensure interactivity, all of these serial operations must happen in the order of milliseconds. Intuitively, this amount of time, which is defined as *interaction delay*, must be kept as short as possible in order to provide a rich experience to the cloud game players. However, there are tradeoffs: the shorter the player's tolerance for interaction delay, the less time the system has to perform such critical operations as scene rendering and video compression. Also, the lower this time threshold is, the more likely a higher network latency can negatively affect a player's experience of interaction. With this is mind, we start our design discussion with delay tolerance.

A. Interaction Delay Tolerance

Studies on traditional gaming systems have found that different styles of games have different thresholds for maximum tolerable delay [4]. Table I summarizes the maximum delay that an average player can tolerate before the Quality of Experience (QoE) begins to degrade. As a general rule, games that are played in the first person perspective, such as the shooter game Counter Strike, become noticeably less playable when actions are delayed by as little as 100 ms. This low delay tolerance is because such first person games tend to be action-based, and players with a higher delay tend to have a disadvantage [5]. In particular, the outcome of definitive game changing actions such as who "pulled the trigger" first can be extremely sensitive to the delay in an action-based First Person Shooter (FPS) game. Third person games, such as Role Playing Games (RPG), and many massively multi-player games, such as World of Warcraft, can often have a higher delay tolerance of up to 500 ms. This is because a player's commands in such games, e.g., use item, cast spell, or heal character, are generally executed by the player's avatar; there is often an invocation phase, such as chanting magic words before a spell is cast, and hence the player does not expect the action to be instantaneous. The actions must still be registered in a timely manner, since the player can become frustrated if the interaction delay causes them a negative outcome, e.g., they healed before an enemy attack but still died because their commands were not registered by the game in time. The last category of games are those played in an "omnipresent" view, i.e., a top down view looking at many controllable entities. Examples are Real Time Strategy (RTS) games like Star Craft and simulation games such as The Sims. Delays of up to 1000 ms can be acceptable to these styles of games since the player often controls many entities and issues many individual commands, which often take seconds or even minutes to complete. In a typical RTS game, a delay of up to 1000 ms for a build unit action that takes over a minute will hardly be noticed by the player.

Although there is much similarity between interaction delay tolerance for traditional gaming and cloud gaming, we must stress the following critical distinctions. First, traditionally, the interaction delay was only an issue for multi-player online gaming systems, and was generally not considered for single player games. Cloud gaming drastically changes this; now all games are being rendered remotely and streamed back to the player's thin client. As such, we must be concerned with interaction delay even for a single player game. Also, traditional online gaming systems often hide the effects of interaction delay by rendering the action on a player's local system before it ever reaches the gaming server. For example, a player may instruct the avatar to move and it immediately begins the movement locally; however the gaming server may not receive the update on the position for several milliseconds. Since cloud gaming offloads its rendering to the cloud, the thin client no longer has the ability to hide the interaction delay from the player. Visual cues such as mouse cursor movement can be delayed by up to 1000 ms, making it impractical to expect the player will be able to tolerate the same interaction delays in cloud gaming as they do in traditional gaming systems. We conjecture that the maximum interaction delay for all games hosted in a cloud gaming context should be at most 200 ms. Other games, specifically such action-based games as first person shooters likely require less than 100 ms interaction delay in order not to affect the players QoE. Recent research using subjective tests have indicated the that this is indeed the case [6].

B. Video Streaming and Encoding

We next examine the video streaming and encoding needs of a cloud gaming system. Cloud gaming's video streaming

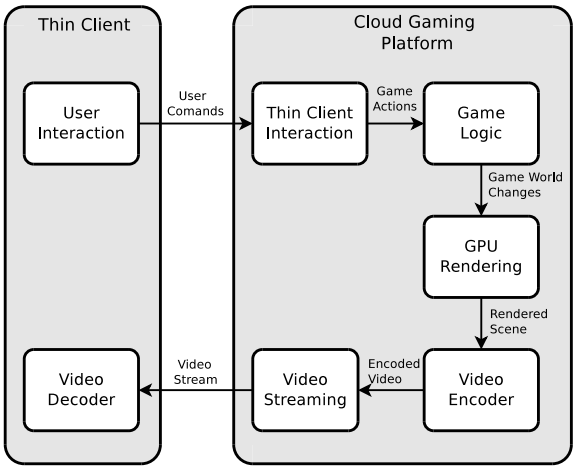


Fig. 2. Framework of a cloud gaming platform

requirements are quite similar to another classical application, namely, live media streaming. Both cloud gaming and live media streaming must quickly encode/compress incoming video and distribute it to end users. In both, we are only concerned with a small set of the most recent video frames and do not have access to future frames before they are produced, implying encoding must be done with respect to very few frames.

However, live video streaming and cloud gaming also have important differences. First, compared to live media streaming, cloud gaming has virtually no capacity to buffer video frames on the client side. This is because, when a player issues a command to the local thin client, the command must traverse the Internet to the cloud, be processed by the game logic, rendered by the processing unit, compressed by the video encoder and streamed back to the player. Given that this must all be done in under 100 - 200 ms, it is apparent that there is not much margin for a buffer. Live media streaming on the other hand can afford a buffer of hundreds of milliseconds or even a few seconds with very little loss to the QoE of the end user.

The sensitive real time encoding needs of cloud gaming make the choice of video encoder of paramount importance for any cloud gaming provider. Currently, the major cloud gaming providers Gaikai and Onlive both use versions of the H.264/MPEG-4 AVC encoder. Gaikai uses a software based approach for encoding where as Onlive is using specialized hardware to compress its cloud gaming video streams. In either case the choice of the H.264 encoder is motivated by the fact that the encoder not only has a very high compression ratio but also that it can be configured to work well with stringent real time demands.

III. Cloud Gaming Framework

Based on the design considerations we have been discussing, we now outline a generic framework for a cloud gaming system. Figure 2 shows the various functions and modules required by a cloud gaming system. As can be observed, a player's commands must be sent over the Internet from its thin client to the cloud gaming platform. Once the commands reach the cloud gaming platform they are converted into appropriate in-game actions, which are interpreted by the game logic into changes in the game world. The game world changes are then processed by the cloud system's graphical processing unit (GPU) into a rendered scene. The rendered scene must be compressed by the video encoder, and then sent to a video streaming module, which delivers the video stream back to the thin client. Finally, the thin client decodes the video and displays the video frames to the player.

To confirm the representability of this generic framework, we have conducted a traffic measurement and analysis from the edge of four networks which are located in the United States, Canada, China and Japan. We recorded the packet flow of both Gaikai and Onlive. After that, we used Wireshark to extract packet-level details, which reveal the existence of thin clients and their interactions with remote cloud servers. We also discover that Gaikai is implemented using two public clouds, namely Amazon EC2 and Limelight. When a player selects a game on Gaikai, an EC2 virtual machine will first deliver the Gaikai game client to the player. After that, it forwards the IP addresses of game proxies that are ready to run the selected games to the players. The player will then select one game proxy to run the game. For multiplayer online games, these game proxies will also forward the players' operations to game servers and send the related information/reactions back to the players. Onlive's workflow is quite similar, but is implemented with a private cloud environment. Using public clouds enables lower implementation costs and higher scalability; yet a private cloud may offer better performance and customization that fully unleash the potentials of cloud for gaming. Hence, we use Onlive in the following measurement and analysis.

IV. Real World Performance: Onlive

Despite some recent financial issues, Onlive was one of the first to enter into the North American market and offers one of the most advanced implementations of cloud gaming available for analysis. A recent official announcement from Onlive put the number of subscribers at roughly 2.5 million, with an active user base of approximately 1.5 million. We evaluate the critically acclaimed game Batman Arkham Asylum on Onlive and compare its performance to a copy of the game running locally. In our analysis, we look at two important metrics, namely, the interaction delay (response time) and image quality. Our hardware remains consistent for all experiments. We run Batman through an Onlive thin client as well as locally on our local test system. The test system contains an AMD 7750 dual core processor, 4 GB of ram, a 1-terabyte 7200 RPM hard drive, and an AMD Radeon 3850 GPU. The network access is provided through a wired connection to a residential cable modem with a maximum connection speed of 25 Mb/s for download and 3 Mb/s for upload. Our system specifications and network connections exceed the recommended standards both for Onlive and the local copy of the game, which ensures

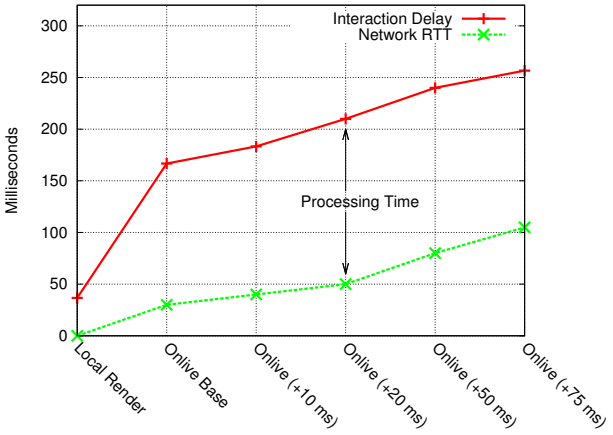


Fig. 3. Interaction Delay in Onlive

Measurement	Processing Time (ms)	Cloud Overhead (ms)
Local Render	36.7	na
Onlive base	136.7	100.0
Onlive (+10 ms)	143.3	106.7
Onlive (+20 ms)	160.0	123.3
Onlive (+50 ms)	160.0	123.3
Onlive (+75 ms)	151.7	115.0

TABLE II
PROCESSING TIME AND CLOUD OVERHEAD

the bottleneck that we will see is solely due to the intervention of cloud.

A. Measuring Interaction Delay

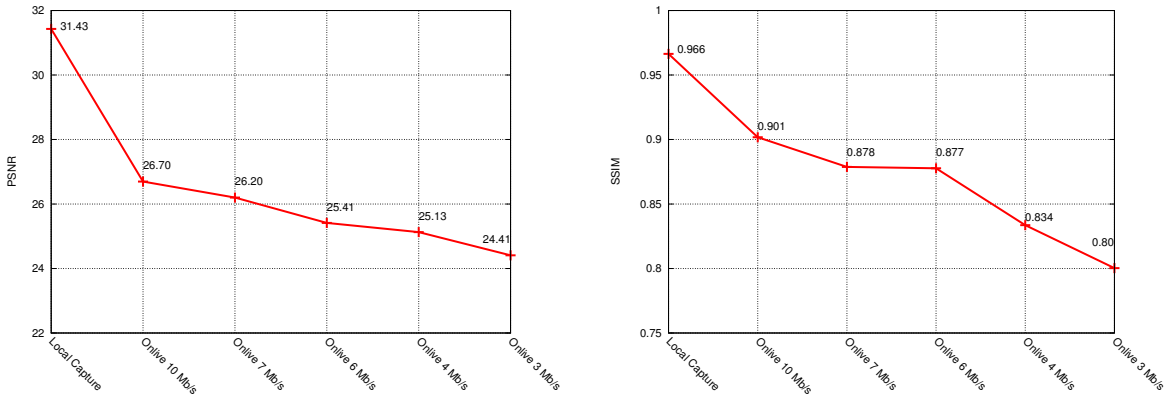
As discussed previously in section II-A, minimizing interaction delay is a fundamental design challenge for cloud gaming developers and is thus a critical metric to measure. To accurately measure interaction delay for Onlive and our local game, we use the following technique. First, we install and configure our test system with a video card tuning software, *MSI afterburner*. It allows users to control many aspects of the system's GPU, even the fan speed. We however are interested in its secondary uses, namely, the ability to perform accurate screen captures of gaming applications. Second, we configure our screen capture software to begin recording at 100 frames per second when we press the "Z" key on the keyboard. The Z key also corresponds to the "Zoom Vision" action in our test game. We start the game and use the zoom vision action. By looking at the resulting video file, we can determine the interaction delay from the first frame that our action becomes evident. Since we are recording at 100 frames per second, we have a 10 millisecond granularity in our measurements. To calculate the interaction delay in milliseconds, we take the frame number and multiply by 10 ms. Since recording at 100 frames per second can be expensive in terms of CPU and hard disk overhead we apply two optimizations to minimize the influence that recording has on our games performance. First, we resize the frame to 1/4 of the original image resolution. Second, we apply Motion JPEG compression before writing to the disc. These two optimizations allow us to record at 100 frames per second, while using less than 5% of the CPU and writing only 1 MB/s to the disk.

To create network latencies, we set up a software Linux router between our test system and Internet connection. On our router we install the Linux network emulator *Netem*, which allows us to control such network conditions as network delay. We determine that our average base-line network Round Trip Time (RTT) to Onlive is approximately 30 milliseconds with a 2 ms standard deviation. For each experiment we collect 3 samples and average them. The results can be seen in Figure 3, where the labels on the Onlive data points indicate the added latency. For example, Onlive (+20 ms) indicates that we added an additional 20 ms on the network delay, bringing the total to 50 ms. Our locally rendered copy has an average interaction delay of approximately 37 ms, whereas our Onlive baseline takes approximately four times longer at 167 ms to register the same game action. As is expected, when we simulate higher network latencies, the interaction delay increases. Impressively, the Onlive system manages to keep its interaction delay below 200 ms in many of our tests. This indicates that for many styles of games Onlive could provide acceptable interaction delays. However, when the network latency exceeds 50 ms, the interaction delays may begin to hinder the users' experience. Also, even with our baseline latency of only 30 ms, the system could not provide an interaction delay of less than 100 ms, the expected threshold for first person shooters.

We next further examine the delay into detailed components. Returning to Figure 3, we define the processing time to be the amount of interaction delay caused by the game logic, GPU rendering, video encoding, etc; that is, it is the components of the interaction delay not explained by the network latency. For example, our locally rendered copy of the game has no network latency; therefore its processing time is simply 37 ms. Our Onlive-base case, on the other hand, has its communication delayed by approximately 30 ms due to the network latency, meaning its processing time is approximately 137 ms. Finally, we calculate the cloud overhead, which we define to be the delay not caused by the core game logic or network latency. It includes the amount of delay caused by the video encoder and streaming system used in Onlive. To calculate this number, we subtract the local render processing time of 37 ms from our Onlive experiment processing time. Table II gives the interaction processing and cloud overhead measured in our experiments. As can be seen, the cloud processing adds about 100-120 ms of interaction delay to the Onlive system. This finding indicates that the cloud processing overhead alone is over 100 ms, meaning that any attempt to reach this optimal interaction delay threshold will require more efficient designs in terms of video encoders and streaming software.

B. Measuring Image Quality

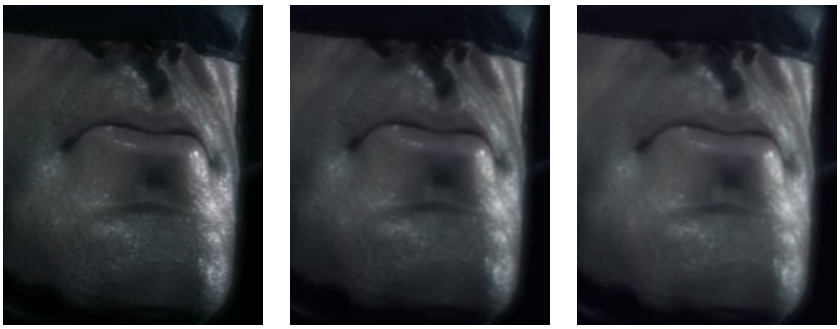
Just as critical as low interaction delay to a cloud game player is image quality. As mentioned previously Onlive uses



(a) PSNR experiments

(b) SSIM experiments

Fig. 4. Onlive comparison



(a) Master Image

(b) Local Capture (PSNR:33.85 dB, SSIM:0.97)

(c) Onlive: 10 Mb/s Connection (PSNR:26.58 dB, SSIM:0.94)

(d) Onlive: 6 Mb/s Connection(PSNR:26.53 dB, SSIM:0.92)

(e) Onlive: 3.0 Mb/s Connection (PSNR: 26.03 dB, SSIM:0.89)

Fig. 5. Image Quality Comparison

a hardware H.264 encoder with a real-time encoding profile, implying the compression will cause some degree of image quality loss. Devising a methodology to objectively analyze the image quality of a commercial cloud gaming system such as Onlive has a number of technical challenges. First, to obtain an accurate sample for the video quality analysis, we must be able to record a deterministic sequence of frames from Onlive and compare it to our local platform. Yet, although the stream is known to be encoded by H.264, the stream packets can hardly be directly captured and analyzed since it appears that Onlive is using a proprietary version of the Real Time Transport Protocol (RTP). The rendering settings used by Onlive are not publicly visible, either. For example, it remains unknown if Onlive has enabled anti-aliasing or what the draw distance is for any game. With these issues in mind, we have determined the following methodology to measure Onlive image quality.

Once again we select the popular game Batman Arkham Asylum as our test game, and we use the same test platform described previously. To mitigate the effect that different rendering settings have on the image quality, we choose the pre-rendered intro movie of the game to record. To improve the

accuracy of our analysis, we unpack the intro video's master file from the game files of our local copy of Batman Arkham Asylum. The extracted movie file has a resolution of 1280 x 720 pixels (720p), which perfectly matches the video streamed by Onlive. We also configured our local copy of Batman to run at a resolution of 1280 x 720 pixels. We configured our display driver to force a frame rate of 30 FPS to match the rate of target video. Next, we configure MSI afterburner to record the video uncompressed with a resolution of 1280 x 720 pixels at 30 FPS. The lack of video compression is very important as we do not want to taint the samples by applying lossy compression.

We then capture the intro sequence of our locally running game and Onlive running with different bandwidth limits. To control the bandwidth, we again use our Linux software router and perform traffic shaping to hit our targets. We test Onlive running from its optimal bandwidth setting of 10 Mb/s gradually down to 3.0 Mb/s. It covers a broad spectrum of bandwidths commonly available to residential Internet subscribers. Before each run, we ensure our bandwidth settings are correct by a probing test. After capturing all the required video sequences, we select the same 40 second (1200 frame) section from each video on which to perform an image quality analysis. We analyze the video using two classical metrics, namely *Peak Signal-to-Noise Ratio* (PSNR) and *Structural Similarity Index Method* (SSIM). The results for PSNR are given in Figure 4a and SSIM are given in Figure 4b, respectively. The PSNR method quantifies the amount of error (noise) in the reconstructed video, which has been added during compression. The SSIM method calculates the structural similarity between the two video frames. As can be seen, our local capture scored a high PSNR and SSIM; however it is not perfect, indicating some difference in the recorded video and the master file. Much of this difference is likely due to slightly different brightness and colour settings used by the internal video player in the Batman game engine. When the local capture is compared to Onlive running at any connection rate, we can see a large drop in terms of both PSNR and SSIM. Since PSNR and SSIM are not on a linear scale, the drops actually indicate a considerable degradation in image quality. Generally a PSNR of 30 dB and above is considered good quality, however 25 and above is considered acceptable for mobile video streaming. Not surprisingly, as we drop our test systems connection bandwidth the image quality begins to suffer considerable degradation as well. With the exception of the 3.0 Mb/s test, all samples stay above a PSNR of 25 dB; so although there is room for improvement, the image quality is still acceptable. Figure 5 illustrates the effect of Onlive's compression taken from a single frame of the opening sequence. As can be seen the effect of compression is quite noticeable especially as the amount of available bandwidth decreases.

V. Conclusion and Further Discussion

This article has closely examined the framework design of state-of-the-art cloud gaming platforms. We have also measured the performance of Onlive, one of the most representative and successful cloud gaming platforms to date. The results, particularly on interaction latency and streaming quality under diverse game, computer, and network configurations, have revealed the potentials of cloud gaming as well as the critical challenges toward its widespread deployment. For a future work we would like to further investigate the effect other network conditions such as packet loss and jitter have on the end users cloud gaming experience.

Cloud gaming is a rapidly evolving technology, with many exciting possibilities. One frequently mentioned is to bring advanced 3D content to relatively weaker devices such as smart phones and tablets. This observation is made even more relevant by the fact that both Gaikai and Onlive are actively working on Android apps to bring their services to these mobile platforms. However, recent large scale research indicates that it is not uncommon to find cellular network connections that have network latencies in excess of 200 ms [7], which alone may already cause the interaction delay to become too high for many games. Seamless integration between cellular data connection and the lower latency WiFi connection is expected, and the switching to LTE may help alleviate the problem. Other potential advancements involve intelligent thin clients that can perform a portion of the game rendering and logic locally to hide some of the issues associated with interaction delay, or distributed game execution across multiple specialized virtual machines [8]. This will likely require creating games specifically optimized for cloud platforms.

Besides software and service providers, hardware manufacturers have also shown a strong interest in cloud gaming, and some have begun working on dedicated hardware solutions to address the prominent issues of cloud gaming. NVIDIA has just unveiled the GeForce grid graphical processor, which is targeted specifically towards cloud gaming systems [9]. It is essentially an all in one graphical processor and encoding solution. The published specification shows that each of these processors has enough capability to render and encode four games simultaneously. NVIDIA's internal tests show that it can significantly mitigate the latency introduced in current cloud gaming systems [10]. It is widely expected that this type of specialized hardware will usher in a new generation of cloud gaming.

References

[1] Onlive. http://www.onlive.com/.

[2] Gaikai. http://www.gaikai.com/.

[3] Engadget. Sony buys Gaikai cloud gaming service for 380 million. http://www.engadget.com/2012/07/02/sony-buys-gaikai/.

[4] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[5] M. Claypool and K. Claypool. Latency can kill: precision and deadline in online games. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, pages 215–222. ACM, 2010.

[6] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 330–335. IEEE, 2011.

[7] J. Sommers and P. Barford. Cell vs. wifi: on the performance of metro area mobile connections. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 301–314. ACM, 2012.

[8] Z. Zhao, K. Hwang, and J. Villeta. Game cloud design with virtualized cpu/gpu servers and initial performance results. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 23–30. ACM, 2012.

[9] Geforce grid. http://www.nvidia.ca/object/grid-processors-cloud-games.html.

[10] James Wang. Nvidia geforce grida glimpse at the future of gaming. http://www.geforce.com/whats-new/articles/geforce-grid.