Overcoming Latency Issues in Streaming Cloud-based Games

Waleed Ahmad
Department of Computer Science
FAST-NUCES
Lahore, Pakistan
l181282@lhr.nu.edu.pk

Muhammad Daud Mazhar
Department of Computer Science
FAST-NUCES
Lahore, Pakistan
l180919@lhr.nu.edu.pk

Sameer Ali
Department of Computer Science
FAST-NUCES
Lahore, Pakistan
l181150@lhr.nu.edu.pk

***Abstract* - The increase in hardware intensive games and graphical rendering has led to the production of high-end equipment capable of running these applications. Such demanding hardware is difficult for users to maintain, especially for those who desire portability and low profile computer system setups. This gives birth to the concepts of cloud gaming. Although cloud gaming is a solution to players who want to access powerful machines capable of running high requirement programs, it faces a lot of issues and suffers from many cons. The underlying research will focus on identifying most of these issues, and will propose current solutions, implemented by most companies today. The results extracted from these solutions will help in proposing a theoretical solution to ensure smooth streaming and running of these games through cloud servers.**

I. Introduction

Cloud gaming, which is based on the concept and implementation of cloud streaming, uses remote servers and the internet to stream video games to the client device. Cloud gaming is also termed as 'gaming on demand', or even 'gaming as a service'. These video games run on remote servers, which are high specification computers, and stream the content directly to a device, such as a mobile phone, tablet, computer or even a TV. The device simultaneously sends the input back to the host server, which is the machine running as well as streaming the game. As a result, the actual gameplay occurs on the server, while the interaction occurs at the client side. The fact that these games run on remote servers, give the whole concept the name of cloud-gaming, like what we see in traditional cloud computing. Similar to the remote desktop concept, each game is active on a server licensed to an authorized user. Registered users can therefore access a high speed computer, usually solely for gaming, to play demanding titles from around anywhere in the world, provided they have a good internet connection. Some popular cloud-gaming setups include Sony's PlayStation NOW, Nvidia's Geforce NOW, Google's Stadia and Vortex. All of these use remote setups to run games and stream them to local machines. In comparison, they may be better or worse than the other in many aspects, but they are based on the same concept - cloud gaming. This concept itself may seem quite simple, however, in the practical application of the cloud gaming, there are many hurdles to overcome such as the setting up of the infrastructure, the global availability of the service, intra-ISP data caps implemented by some, the quality of experience (QoE), et cetera. The main focus of this paper is to identify the factors that cause latency in experiencing the actual game over the cloud, what users of the system may have to deal with at the moment, and what the possible solutions to the problems experienced by the set up are. Primarily, we will discuss the network congestion issues, the inter-player delay optimization (IDO), which in fact, comprise the latency issue being addressed by this paper, and shed light on the probable solutions of overcoming them.

II. Architecture & Framework

Cloud computing is achieved by implementing the 'gaming as a service (GaaS)' model. There are multiple types of these models, such as the remote-rendering GaaS (RR-GaaS), the local-rendering GaaS (LR-GaaS) and the cognitive resource allocation GaaS (CRA-GaaS).

In the *remote-rendering GaaS model*, an encoder at the server renders the game data and transmits it to the client, where the sent stream is decoded and displayed to the user. At the same time, the client inputs are recorded and sent to the host or the server where the encoder lies. These inputs are mapped to the game simultaneously. The advantage of this model lies in the fact that less powerful machines are able to allow the user to play high-end games, only under suitable conditions like a capable network. An example of service using this model is Google Stadia and Nvidia GeForce Now.

Remote-rendering model is the main architecture for this research.

In the *local-rendering GaaS model*, the rendering instructions are received by the client from the host which are used to render the game frames for the user to view. This architecture requires graphical power at the client level for the purpose of rendering, but low server load.

In the cognitive *resource allocation model*, the partitioned data is kept by the client side for rendering. Unused data is discarded and only the needed components are loaded by the client as the application progresses. In this architecture the load on the server side is reduced, while the load on the user side is maintained by discarding unused components.

Figure 1 gives a detailed description of how the cloud gaming framework works.

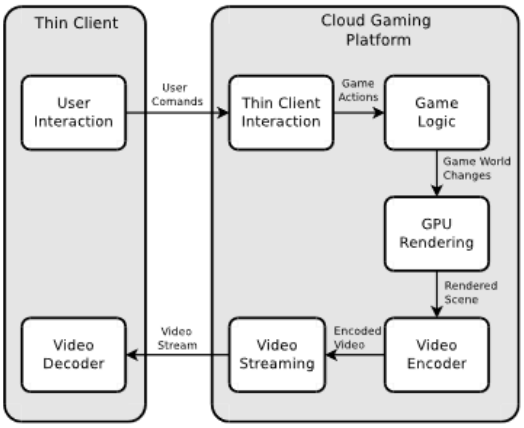

Figure 1: Cloud gaming platform framework.

We have discussed what the actual models of cloud gaming architecture are, we now outline how the framework implemented by these models work. A client is usually an internet capable device such as a computer, phone or even a TV. These devices read inputs through peripherals such as keyboards, or gamepads to read inputs at client level and send them to the cloud service, where they are read and mapped to the game. This results in the game reacting to the inputs and changes accordingly. The hardware at the cloud level processes these changes and generates a video stream of the rendered game scene which is sent back to the client encoded. The client decodes this video stream to reveal the actual game to the user. In short, it is a two-way communication as each transfer. Let's take Google Stadia's framework for example. The system requires a Stadia capable device (an Android, an iOS, or a chromecast with the Stadia application installed), a Stadia controller, and an internet connection. The controller sends it's input directly to Google's server rather than to the device through Wi-Fi. Although most devices still allow the input mappings to be sent to the device directly. The input mappings are received by Google servers and the changes are then reflected on the devices' screen.

III. Identifying the Major Problems

Like already discussed, the typical framework of cloud gaming makes use of Virtual Machines (VMs). At the client end, the VM appears as the client machine rendering the game graphics, while at the other end, it acts as a remote cloud server to host the game.

The biggest challenge for cloud based gaming, as compared to traditional online gaming, is maintaining the game session with acceptable responsiveness and graphics rendering despite the underlying issues created by the network delay or processing cycles. Existing cloud gaming platforms are primarily based upon video streaming, whereby the clients only decode the rendered information processed by the cloud servers. Naturally, the video streaming limitations accompany, which include (1) high bandwidth consumption, (2) limited scalability and (3) little room for optimization [1].

For Massively Multiplayer Online Gaming (MMOG), the latency is further affected by the addition of the MMOG Server. A player's inputs need to be forwarded to the cloud server in addition to the MMOG server and then another cloud server before these can be transferred onto another player's device. This produces extremely high latency issues which will be referred to as Interaction Delays within this paper. [2]

A. Interaction Delay Tolerance

Different game genres are prone to different thresholds for bearing interaction delays. Most First Person Shooter (FPS) games, like Counter Strike, require minimum delays in order to ensure a playable stage. The maximum delay that an average gamer can tolerate for such FPS games before the QoE starts degrading is 100 ms. Third person or Role Playing Games (RPG) have a higher degree of tolerance as they usually integrate invocation phases where players do not expect almost instantaneous rendering of their actions. However, interaction delays exceeding about 500 ms are noticeable and can degrade the QoE for such games. Lastly, Real Time Strategy (RTS) games show the most leverage for such delays, where maximum tolerable delays can reach as high as 1000 ms, without even being noticed by players. Table 1 summarizes the maximum threshold for Interaction Delay Tolerances of different gaming styles [3].

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

Table 1: Delay Thresholds for Different Game Types

B. Video Streaming and Encoding

Cloud's game streaming is similar to any video streaming service, where both must ensure rapid encoding/compression of input video and distribute it over the network to all the requesting client systems. But unlike live media streaming, cloud streaming differs in that it has virtually no buffer to store the frames. This is because let's say, the user registers a command to his machine, the inputs must traverse the internet, reach the cloud server hosting the game logic for processing, and then, the rendered scene is compressed by the video encoder and sent back to the client for rendering after decoding. This complete process is preferred to be completed under 100-200 ms. As previously mentioned in Table 1, this delay is exceeding tolerable levels, and therefore, does not leave us with enough room for further buffering of the frames.

To further demonstrate the two major problems identified, we will consider an experiment in order to distinguish the performance of a game played on a local machine versus the same game played on a major cloud gaming provider, Onlive, which uses a version of H.26/MPEG-4 AVC encoder, coupled with specialized hardware for compression, implemented under a private cloud configuration for better performance. The local machine's hardware contains an AMD 7750 dual-core processor, 4 gigs of RAM, powered by AMD Radeon 3850 GPU. The internet connection is established via an ethernet cable with downloading speeds as high as 25 Mbps and 3 Mbps uploading speed. All these configurations on both the systems exceed the recommended system specifications for running the acclaimed game *Batman Arkham Asylum*. These specifications were kept as control in order to ensure that any bottleneck we observe is solely due to cloud intervention. The results are shown in Figure 2 and the evaluations drawn from these results are summarized in Table 2.

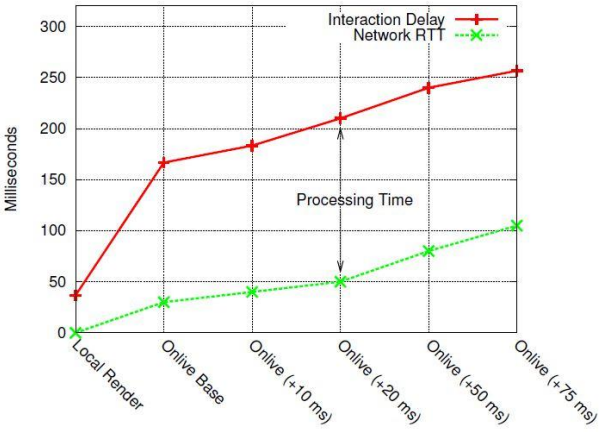

Figure 2: Interaction Delay in Onlive

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 2: Processing Time and Cloud Overhead

Now that we understand the underlying problems in using cloud gaming services, we will look at the related work and proposed solutions to overcome these latency challenges in order to establish a seamless gaming experience.

IV. Literature Review & Current Solutions

Cloud gaming is quite similar to approaches used in remote rendering, which allows distinction in rendering operations from the client to cloud servers. Aforementioned, the higher the geographical distance between two connecting client/server systems, higher the latencies or inter-player delays. Still young, but we have multiple approaches to cater for these issues in order to optimize the cloud gaming experience even for highly delay sensitive games or MMOGs. We will discuss each of the several approaches used already and try to extract the best features out of each in order to frame a hybrid solution that could transform the gaming experience. Since latency issues are based on several factors, we can divide these issues into two primary categories:

(a) interaction delays

(b) graphics/frames rendering quality

The paper, *Inter-player Delay Optimization in Multiplayer Cloud Gaming,* by Yuchi Chen et al [4] and *Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud* by Richard Süselbeck et al [2] will primarily focus on minimizing the interaction delays. The first paper poses an optimistic algorithm for the relocation of servers in order to minimize the geographical separation between the host servers and connecting clients. The latter paper researches the possibility of co-existence of cloud and MMOG servers, which again is a similar approach as the former one, in order to optimize the input delays.

The graphics rendering quality optimization techniques are discussed in *LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming* by Xiaofei Liao et al [5]. This paper attempts to explore an approach that tries to compress the graphics so that efficient transmission of rendered frames could be made possible over the cloud. *Optimizing next-generation cloud gaming platforms with planar map streaming and distributed rendering* by Pin-Chun Wang et al [1] provides an in-depth detail of how Planar Map Streaming can be used via distributed rendering to optimize cloud gaming performance. Each of these papers are reviewed below for further insight on the details of each approach.

A. Inter-player Delay Optimization in Multiplayer Cloud Gaming

This research [4] talks about the Quality of Experience in terms of interaction delay experienced by the users at the local end of the system. The research talks of the difference in delays of users interacting with the cloud server, or the Inter-player Delay and identifies the issues it produces, along with the possible solutions. Inter-player Delay is a threat to the quality of cloud-gaming as it removes the fairness in competitive games, mainly FPS games (e.g, Call of Duty, Battlefield, Counter-Strike) and fighting games (e.g, Mortal Kombat, Tekken). The solution to inter-player delay is the method of Inter-player Delay Optimization which is aimed at reducing this delay. Although the IDO method seems to be the solution to resolving this delay, it has problems of its own. Firstly, it has its own complexity as it is aimed at optimizing both network latency and processing latency in between all the users interacting with one another through cloud set-ups. Secondly, the Inter-player Delay Optimization problem requires the response latency to be acceptable such that the quality of the gameplay is not compromised. The third challenge that IDO puts forth is scalability of server farms and data centers. The article further explains the ways to tackle the problems and challenges produced by IDO.

The research proposes the solutions in two parts. The first is to find the best placement of the players' virtual machine, which hosts the server. These placements of the virtual machines refer to minimizing the response delay experienced by the players. The second part of the solution is to adjust the placements of virtual machines such that the overall delay is reduced. The algorithm in the first step first generates a circular corvex set for each data center. This set includes all the accessible virtual machines. Then A relative covering cost is calculated for each of each set. This way the convex set with the least cost is chosen, and the virtual machine is omitted from all other sets, leaving only one player to be served at a time in each iteration.

In the second part of the solution, or the adjustment phase, a local search is implemented to minimize inter-player delay. In this strategy, the virtual machines of players are moved from data centers depending on their delays. For a player with high absolute response delay, the virtual machine is moved to a more suitable data center. It subsequently checks if the response delay is reduced, or is under a certain threshold. Not only does this reduce absolute response delay, it maintains the delay under a threshold such that QoE is not compromised. A brief idea of this VM and DC setup is given in Figure 3.

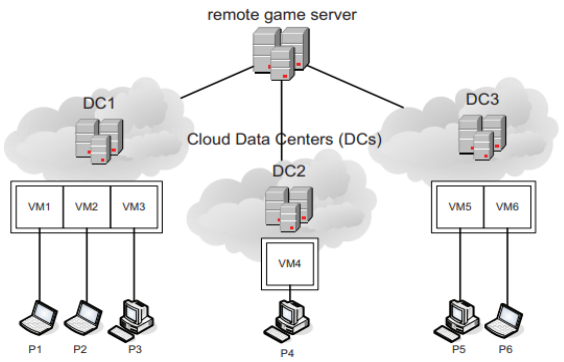

Figure 3: Cloud gaming network with VMs and DCs

The research claims that the proposed algorithm can reduce about thirty-percent of the inter-player delay. The evidence proves their case as it provides real-world data sets, as they point out how it can also reduce the number of data centers along with reducing overall player to player delay.

B. Peer-to-Peer Support for Low-Latency Massively Multiplayer Online Games in the Cloud

Massively Multiplayer Online Games [2] (MMOG) incurs a delay penalty because in addition to the two cloud servers interacting with their respective clients, an MMOG server is needed. As a result, the number of hops for an input to reach from player 1 to player 2 becomes at least four. To further clarify this, we can refer to Figure 4a which shows the intermediate hops between two peer clients trying to connect within the game. Each player's inputs need to traverse at least four major hops in an MMOG environment, all of which add to input

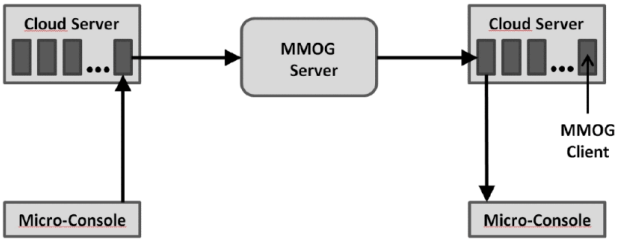

Figure 4a: Client-Server Based MMOG Model in Cloud Gaming

delays. In case of such MMOG platforms, the games are usually very delay sensitive, and therefore, make the games, where minimum response times are vital, highly unplayable.

We observe that if all MMOG players were connected to the same cloud server, the MMOG server could be co-placed with the cloud server. By doing so, the additional input delays could be eliminated. But there is a grave problem with this approach. For MMOGs, the connecting clients are geographically located throughout the world. Hence, a need arose to devise a method to enable the co-location of MMOG servers within the cloud using multiple geographically distributed servers. In order to do so, a co-located cloud/MMOG server caters for handling the game client as well as an instance of the peer-to-peer middleware for each end client that connects to it. The middleware itself keeps no information that the peers are running on separate server centers instead of individual client systems. The middleware is responsible for direct updates transmission between peers, therefore, reducing the burden from the cloud server. These updates, as a result, require only two hops instead of four for updates transmission.

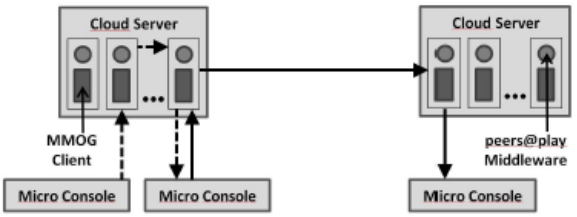

Figure 4b: P2P based MMOG in Cloud

This model reduces the number of hops required to update the instance of a player that may be running on the same or different server center. If both players are connected to the same server, the update can reach the other player in two hops. These hops are increased to three if the other player is connected to some other server center as the update would need to be sent over the Internet. The model presented in Figure 4a uses at least four hops in case both the clients are connected to the same server center. In the modified model, Figure 4b, the maximum number of hops needed are reduced to at max three. Although it is still significant, it is still better than our previous model. Also, it distributes the MMOG server load by dividing the traffic into multiple server centers.

C. LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming

We have explored the existing strategies that pose as solutions for reducing latency due to interaction delays, we now try to observe the rendering element in the architecture. Remote rendering, as we already know, implements the graphic rendering at the server side where the graphics once rendered are converted into video streams and encoded to be sent to the client side. This task puts burden at the server side, and is very less scalable due to this fact. Although the encoded video stream is good for the client as no demanding task is to be done by the client, the issue of huge processing at the server is still an issue. This research [5] focuses more on the local rendering side, where it comes up with ways to cater the problems caused by local rendering and in turn making it an effective solution for the constraints that cloud gaming involves.

We know that in local rendering, the graphic intensive work is unloaded at the client side to be rendered, which causes ever more issues related to bandwidth due to the amount of data. However, the fact that the graphics are offloaded at the client side puts the server at ease allowing higher scalability. At the same time, it requires the client to be capable of even rendering the data.

Keeping all these constraints in mind, the main goal of this research is to introduce the concept of LiveRender. LiveRender is a system which helps in optimizing the datastream so that the bandwidth stress is reduced with the help of something called compressed graphics streaming. The basic idea of compressed graphics streaming is that the graphical data is compressed and cached on both the client and the server. LiveRender uses this phenomenon to reduce not only the bandwidth consumption but also the response delay. LiveRender offers many compression techniques or algorithms to compress graphical data.

Now that we know what LiveRender is, we move on to how it works? Once the server runs the application it is intended to run, it uses LiveRender to compress the data by using 3D Command Wrappers. These allow the data to be cached and compressed into command or geometry. At the client level, something similar happens where the data is decompressed and cached as well. As soon as the server reads a user input, it generates commands to send to the client in parallel. These commands are executed and then rendered at the client until the server stops sending them. Furthermore, LiveRender uses interframe and intraframe compression algorithms to compress and decompress the graphical models. Paired with the cache mechanism which involves caching all three graphical data (graphic command, textures and geometry), the overall network traffic is reduced. It should be noted

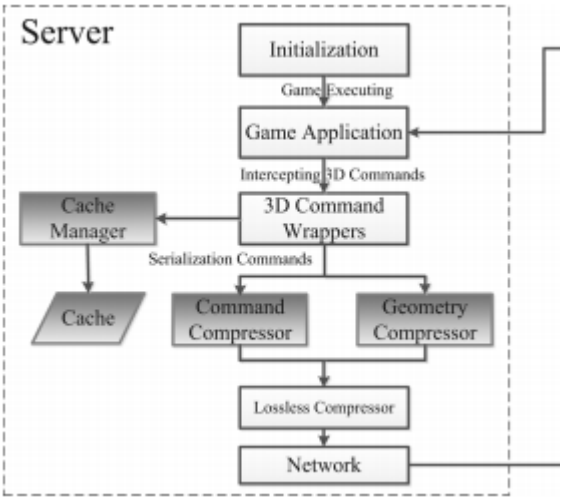

Figure 5a: LiveRender Architecture Server side.

from the above working that the whole system runs in a pipelined manner. Although there is a probability of further optimization of interframe and intraframe, the whole system of LiveRender which uses compressed graphic streaming is seemingly better than raw graphic streaming. Pictorial representations of the model's server and client are given in Figure 5a and Figure 5b.

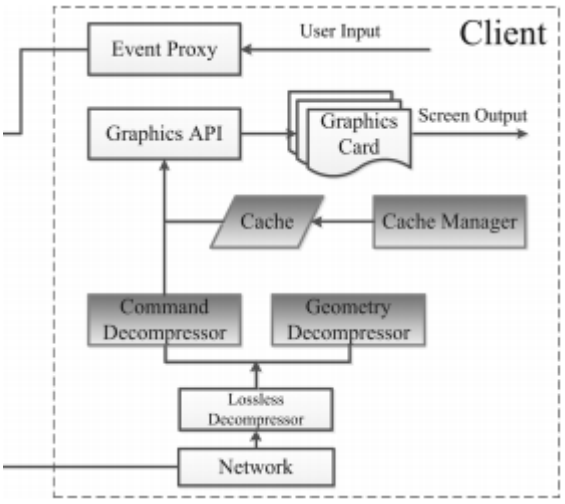

Figure 5b: LiveRender Architecture Client side.

D. Optimizing Next-Generation Cloud Gaming Platforms with Planar Map Streaming and Distributed Rendering

This paper has delved into an unexplored topic of studying compression of 2D planar maps to develop a next-generation cloud gaming platform. This enables them to yield a distributed rendering platform that converts the pipeline of 2D planar maps into server and client pipelines. Planar maps were first introduced by Baudelaire and Gangnet [6] which are fundamentally 2-dimensional objects with an arbitrary complexity.

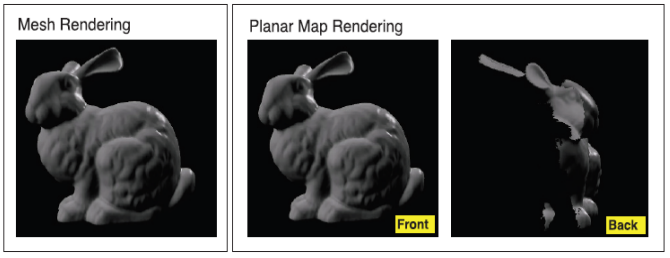

Figure 6: Traditional 3D Mesh vs Planar Mapping Rendering

Planar maps are vector images which makes them optimal in that they can be used to scale for ultra high resolution. Along with that, these vectors are concise which makes them preferable for efficient streaming. Figure 6 can be used to understand why planar maps are the better choice. On the left side, we have our conventional 3D rendering pipelining approach which renders everything, and then uses z-buffering to buffer or

remove the hidden layers on runtime. In contrast to this, the planar map rendering (right side) cleverly manages to render only the visible images. This drastically reduces the depth complexity to one, and makes it feasible for streaming over the cloud.

This research further builds on two components, one being the rendering of the planar maps and the second being the implementation of planar maps in a distributed environment. In order to achieve a concise vector image in planar mapping, multiple steps are followed in order.

(i) Silhouette Detection: In this step, a hash table stores all the edges in a 3D scene. Then, the silhouette edges are detected by distinguishing whether the z-component in the corresponding face normals have opposite signs. Such an edge shared by more than two triangles is then marked as a silhouette.

(ii) Silhouette Clipping: This is the procedure whereby each triangle is clipped using the detected silhouettes. Non-overlapping triangle-silhouette pairs are removed, and each triangle is clipped using the list of shared edges.

(iii) Triangle-Triangle Occlusion: In this step, a lightweight centroid test is incorporated to detect visibility. The triangles that share some overlapping centroids are discarded.

(iv) Vector Rendering: In this final step, barycentric interpolation generates necessary vertex attributes. This vertex data is passed onto the GPU for rendering and a planar mapped image is produced. Since the concept of planar maps was not used for distributed systems, this paper devised a strategy to adopt this rendering method in cloud computing. The rationale behind this is to allow the client server to render light weight vectors.

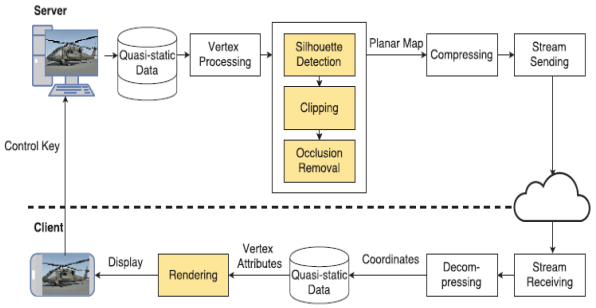

Figure 6: Revised Planar Map Rendering Pipeline for Cloud Gaming

At the server side, the 3D object models are mapped onto 2D planes as discussed formerly. Then, these planar maps are compressed extensively in order to reduce the bandwidth required for transmission.

V. Our Proposed Solution

By now, we have seen the various approaches under interaction delay optimization and graphics rendering. All of these approaches provide better results than previous cloud gaming models. However, these approaches are restricted to using cloud servers. A relatively new concept of Fog Computing can be integrated into cloud gaming platforms. Fog computing allows the transfer of cloud services to the network edge, which allows significant decrease in delays. While this technology is already configured for IoT applications that cater for a relatively fewer number of applications per fog device, a hybrid of fog and cloud computing can be further researched to develop a next-gen gaming platform. A LAMP algorithm, i.e. Latency Aware Module Placement is an exploratory algorithm that attempts to minimize the distance between hosts of modules [7]. The details of this algorithm are discussed in detail but are limited to IoT applications. A problem with this approach is that as the number of devices per fog device increases, the performance drops drastically. Such a platform is not suitable for managing especially MMOGs where the number of client devices is huge. If such a topology can be configured for cloud gaming, especially where the number of connecting clients is not significantly high, it can greatly reduce the latency issues. The scalar mapping technique discussed in [1] is the best graphics compression and transmission system that allows for better perceptual video quality by up to 0.14 SSIM, and even supports extremely high resolutions like 4K video. The pipelining technique enables fast transmission with input delays as high as only 0.83 milliseconds. If we combine this technique for graphics compression and pipelining coupled with the benefits obtained by fog computing in IoT applications, even better performance can be achieved.

VI. Conclusion & Future of Cloud Gaming

We have explored various algorithms and design choices in order to develop an efficient system for cloud gaming interactions. These are categorized primarily into interaction delay optimization and graphics compression and rendering. Interaction delay optimization techniques help to reduce the geographical placement of hosts connecting to the cloud services. The graphics compression algorithms attempt to reduce the image sizes for better performance rendering the game frames over the cloud with minimal network bandwidth

requirements. As proposed, a hybrid of fog-cloud computing can be explored to reach an optimal combination where the high scalability advantages of cloud computing and low latency benefits of fog computing can be maximized. This would certainly require exhaustive research and experimentation which is yet to be explored. In any case, the future of cloud gaming is promising since each year, highly resource intensive games are developed, and the need for utilizing high-end hardware to run these games call for cloud technologies to be well-developed.

VII. References

[1] P. Wang, A. I. Ellis, J. C. Hart and C. Hsu, "Optimizing next-generation cloud gaming platforms with planar map streaming and distributed rendering," *2017 15th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2017, pp. 1-6, doi: 10.1109/NetGames.2017.7991544.

[2] R. Süselbeck, G. Schiele and C. Becker, "Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud," *2009 8th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2009, pp. 1-2, doi: 10.1109/NETGAMES.2009.5446229.

[3] R. Shea, J. Liu, E. C. -Ngai and Y. Cui, "Cloud gaming: architecture and performance," in *IEEE Network*, vol. 27, no. 4, pp. 16-21, July-August 2013, doi: 10.1109/MNET.2013.6574660.

[4] Y. Chen, J. Liu and Y. Cui, "Inter-player Delay Optimization in Multiplayer Cloud Gaming," *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 702-709, doi: 10.1109/CLOUD.2016.0098.

[5] X. Liao *et al*., "LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming," in *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2128-2139, Aug. 2016, doi: 10.1109/TNET.2015.2450254.

[6] P. Baudelaire and M. Gangnet, “Planar maps: an interaction paradigm for graphic design,” in Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI’89), 1989, pp. 313–318.

[7] Z. Rezazadeh, M. Rezaei and M. Nickray, "LAMP: A Hybrid Fog-Cloud Latency-Aware Module Placement Algorithm for IoT Applications," *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, pp. 845-850, doi: 10.1109/KBEI.2019.8734958.

[8]https://www.academia.edu/48202839/Network_Traffic_Adaptation_for_Cloud_Games

[9]https://www.academia.edu/35273369/Cloud_Streaming_on_Online_Gaming

[10]https://cloud.google.com/architecture/cloud-game-infrastructure

[11]https://www.sfu.ca/~rws1/papers/Cloud-Gaming-Architecture-and-Performance.pdf