Managing Cloud Rendered Applications with Public Cloud Providers

Anubhav Guha

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 31.12.2021

Supervisor

Dr. Matti Siekkinen

Advisor

Dr Matti Siekkinen

Aalto University
School of Electrical Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Anubhav Guha

Title Managing Cloud Rendered Applications with Public Cloud Providers

Degree programme ICT Innovation

Major Cloud Computing Services **Code of major** SCI3081

Supervisor Dr. Matti Siekkinen

Advisor Dr Matti Siekkinen

Date 31.12.2021 **Number of pages** 45 **Language** English

Abstract

This project is a detailed study of designing and development of a middle-ware application that interfaces with public cloud providers and remotely deploys cloud rendered applications into virtual environments acting as a server and streams the video into client application, that can be accessed from external devices and the obtained results of the experiments have been used to explore the user experience and advantages of such a system over traditional hardware rendering.

Keywords Cloud computing, cloud gaming, public cloud, application, game server, video streaming, instances, devices, platform, graphics, lambda, virtual machine

Preface

I want to thank Dr. Matti Siekkinen as my instructor for good guidance.

Otaniemi, 31.12.2021

Anubhav Guha

Contents

Abbreviations

A/V	Audio and Video
HD	High Definition
QA	Quality Analysis
QoE	Quality of Experience
QoS	Quality of Service
CRM	Customer Relationship Management
WiMAX	Worldwide Interoperability for Microwave Access
VM	Virtual Machine
PC	Personal Computer
CPU	Central Processing Unit
RAM	Random Access Memory
SSD	Solid State Drive
OS	Operating System
CRON	Command Run On
API	Application Programmable Interface
JSON	JavaScript Object Notation
SDK	Software Development Kit
HTTP	Hypertext Transfer Protocol
REST	Relational State Transfer
TCP	Transmission Control Protocol
IP	Internet Protocol
RDP	Remote Desktop Protocol
P2P	Peer to Peer
CLI	Command Line Interface
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
AMI	Amazon Machine Image
EFS	Elastic File System
EBS	Elastic Block Storage
S3	Simple Storage Service
SNS	Simple Notification Service
SES	Simple Email Service
A2A	Application To Application
A2P	Application To Person
IAM	Identity and Access management
GCP	Google Cloud Platform
VNC	Virtual Network Computing
GA	Gaming Anywhere open-source platform
MMOG	Massively Multiplayer Online Game

1 Introduction

In the last decade, in almost every field, companies have changed their way of operations with the adoption of cloud to revolutionize Customer Relationship Management solutions, listening to music, streaming movies and indubitably disrupting the gaming industry. Cloud technology innovation has had a vehement effect on gaming while completely evolving the business model of several companies that have migrated from traditional practices of using local or on-premise computing systems to an unreservedly virtual environment where every necessary aspect of gaming, may it be provisioning of resources, rendering of graphics, handling execution overload, even provisioning a virtual network, has been taken care of by prominent public cloud providers.

Cloud computing serves as the key to an entire new scenario in gaming industry where companies get to deliver more value to the end user by providing an improved user experience in terms of quality, performance, availability and cost effectiveness. Offering these on-demand services to the end user using desktops, laptops, gaming consoles and all types of heterogeneous clients push the gaming service providers to adopt highly powerful cloud servers capable of sustaining such gaming applications that require huge amount of processing resources. As it was predicted in [1], various companies like Ubitus[6], Gaikai[5] and On-Live[4] have reached billions of dollars in net-worth, after seriously considering the opportunities offered by Cloud as the ubiquitous solution for gaming. With the availability of a fast broadband internet access that supports reliable core networks to connect a huge number of end users to the virtual gaming servers has been a prime trend setter for end users to migrate from native gaming to cloud gaming. The responsibility of provisioning the processing power to render a game has shifted away from the devices of the user into a cloud data center that delivers the entire gaming experience as frames inside a high definition video stream.[2]

Generally, the involvement of an internet connection is already a main requirement in traditional native gaming that has multiplayer mode - several users with their client applications are connected to one central server that hosts and controls the game environment. In this case, the amount of data that is exchanged over the network between the clients and server is significantly small as compared to the data that is exchanged between the clients that are connected to a virtual gaming server that deploys the entire gaming application as one program and sends the video output to the clients, while receiving instructions or commands from the user. Hence, there is a huge difference between conventional Online Gaming and Cloud Gaming when it comes to network QoE as in conventional Online Gaming, the user experience is generated using the hardware of the client device from the side of the end user. Therefore, network speed or availability neither plays a deciding factor nor has any influence in the user experience. Whereas in Cloud Gaming, the presentation gets delivered to the end user as a video stream that occupies huge spaces of network bandwidth in order to deliver the entire video output and the streaming of High Definition video gets easily affected by the first signs of network problems such as, low bandwidth, socket errors, loss of connectivity including delay in packet reception that falls under the responsibility of Internet Service Provider and cannot be controlled or managed by the end user.[3]

Among the many challenges, this is the major challenge faced by Cloud Gaming in terms of network that needs to be tackled in a smarter way in order to deliver this service in the highest quality expected by end users. Unlike conventional methods of video streaming, Cloud Gaming requires low-latency as well as a constant down-link bandwidth that is relatively much higher than that of common broadband networks. For e.g., first-person-shooter games like Counter-Strike, Call of Duty, Medal Of Honor along with third-person-shooter games like Max Payne, GTA, Prince Of Persia including the most popular among racing games: Need For Speed, that use rendering interfaces such as OpenGL and Direct3D, require extremely high request and response rates between the user and application. Therefore, making it not so ideal for Cloud Gaming, in which network issues might delay the client requests that in turn would certainly cause time lags in the synchronization between the game events and user instructions, thereby reducing the overall QoE. In case of Counter-Strike, the player would be targeted by enemy players before the player notices anyone approaching, only because the video frames might make a nano second delay in reaching the client, before user gets to react to such an event. This allows game developers to design and create a whole new category of games that were never designed before to be played over the Internet and consequently deliver an altogether new user experience.

In order to have a better understanding of how cloud computing plays a major role in the evolution of gaming, this project includes a detailed study into the current scenario of traditional gaming and it's present day standing at par with cloud solutions that offer a different way of hosting and consuming gaming services.

In addition, the below tasks and experiments have been performed and documented that offers an independent insight into the scope of cloud gaming: -

– development of a web application that interfaces with public cloud providers.

– building of api services that creates communication channel between web application and cloud provider.

– building serverless functions that act according to the instructions sent from web application for manipulation and monitoring of cloud resources.

– creation of virtual instances that hosts both the gaming application and video streaming server.

– running streaming client application on mobile devices and evaluate the factors that affect user experience.

Based on the independent research, hands-on experience gained from the development and experimentation of the above modules, considerable amount of data has been gathered to provide an argument that significantly puts cloud solutions for mobile gaming, on a profoundly advantageous position when seen from the perspective of the end user.

2 Background

Cloud computing with its powerful data centres deployed all over the world and the provision of elastic resources, has spun off several new opportunities for new and existing applications starting from File Sharing, document synchronization to media streaming. This has brought cloud platforms tremendously forward in terms of usability and efficiency. Cloud data centers have been strategically placed in different locations in order to reduce latency of user access by computational offloading.[7]

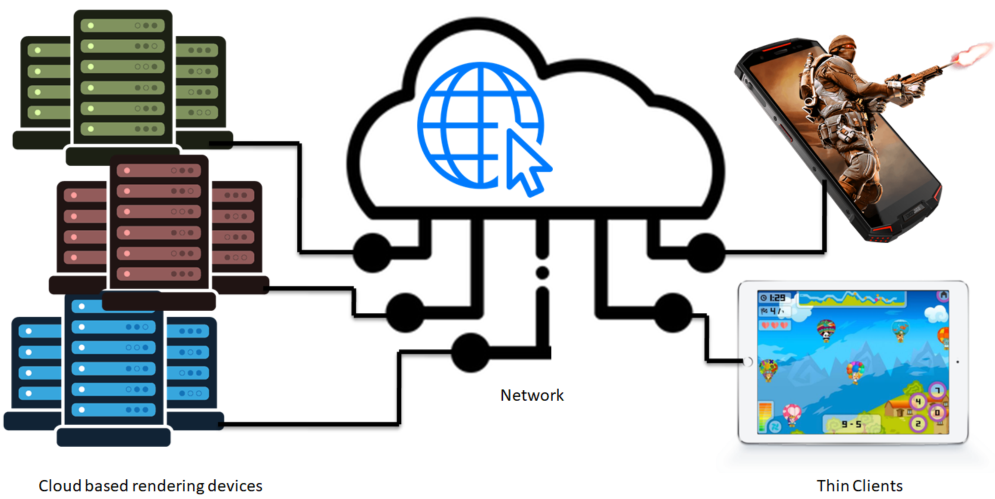

Figure 1: Cloud Gaming Overview

In this solution, users are no longer required to run the games after purchasing expensive hardware, but they could easily get the ready made output from a cloud computer and get the output streamed in any client device capable of playing HD video, as shown in Figure 1. Talking about client devices, even a basic low-end mobile phone that does not support HD video could be used for streaming by reducing the resolution that in turn requires significantly lesser bandwidth to transmit the video and lesser resources are consumed to decode the incoming video stream.[3]

In order to leverage this solution as the most feasible solution, these two objectives should be satisfied: -

1. Reduction of hardware investment

2. Increment of gaming QoE

Since a satisfactory level of QoE requires the processing power of high-end hardware, thereby increasing the setup costs and financial burden, while using low-end hardware might lead to compromise in quality, the prime challenge is to find a suitable tradeoff between the above two objectives that contradict each other.[1]

However, cloud gaming is still in its early stages with the most amount of research being done only in the recently passed few years. Therefore, significant practical and theoretical challenges still remain to be tackled in order to promote a widespread deployment.

Before cloud gaming came into the picture, video streaming of a desktop screen over a network was already a common practice for users who used to exploit softwares like TeamViewer[95], LogMeIn[96], Microsoft Windows Remote Desktop[97]. These softwares have a lot in common with cloud gaming platforms as the users get a similar live downlink containing the video of a different computer screen and an uplink of the response commands sent from the user who is viewing the streamed video. However, cloud gaming platforms differ in terms of the underlying mechanism of how the remote desktops work including the availability of advanced algorithms in a cloud gaming platform that adjusts the video bitrate according to network bandwidth.

Therefore, cloud gaming is a relatively recent paradigm[31] that has received prominent attention in the past decade while spinning off several research projects dedicated to the continuous improvement of this service, consequently setting new standards in the gaming industry having a bright scope in the future envisioned by tech-giant companies like SONY who recently acquired Gaikai which is one of the most admired cloud gaming platforms in the market[98]. Although the work done by cloud gaming platforms is similar to a live video streaming, there are several variations in performance depending on what gaming platform we are using, what genre of games are being played, what operating system is running the game and what are the resource availabilities of hardware acceleration (GPU) provided by different public cloud providers. In the research experiments done in [11], it has been found that On-Live has taken the QoS to a different level by implementing a resource provisioning strategy that is based on the game genre. It has also been observed in the study [12] that the QoS is extremely sensitive to time delays caused by heavy traffic in network that has been mitigated upto a level using a WiMAX[17] network so that the latency values are low enough to almost call it a smooth gaming experience. Considering network conditions as the prime factor on which both the QoS and QoE depend, it has been found in [13] that different cloud gaming platforms generate different densities of network traffic while the network bandwidth consumed varies from 2 Mbps to 14 Mbps. Therefore, network speed affects QoS and QoE much lesser than that of the network problems such as connectivity issues, socket errors, no IP allocation, etc., that makes cloud gaming a much feasible option for gamers in return for its cheap pricing, utmost convenience and ease of use.

Furthermore, the overall process of how cloud gaming platforms work could be directly compared to live video streaming that has maximum data in downlink, keeping the only difference in the uplink that is not present in live video streaming, but present in a cloud gaming platform having very minimum data being sent to the remote gaming computer. This points the finger at the output video that plays the most important role in QoE as the output video quality decides whether the user is satisfied with the QoS or not. Considering the expectations of a regular gamer, it is always expected that every game would have immersive graphics that would attract gamers and capitalize on their interest of playing such a game that would have rich graphics and deliver a cinematic QoE. This would require a GPU to process the graphics rendering and video encoding along with the CPU that takes care of processing the gaming application. Introducing a GPU would not only increase the graphics quality but would also reduce workload on the CPU by avoiding bottlenecks that occur due to running the gaming application and video encoding simultaneously. Cloud computing boasts on the process of virtualization and distribution of shared virtual resources that makes it commodious to provide a virtualized GPU in order to support highly resource demanding games like Call of Duty, Assasins Creed, Command and Conquer, etc. The video output of a heavyweight game requiring high availibility of graphics resources, would have higher bitrate and the transmission would require a fast internet connection having a larger bandwidth. Even after considering all the losses and delays, the main challenge is to find out which game is overall playable after bearing in mind the user tolerance for lags and glitches. This would help categorizing all the games based on their playability and user tolerance that would bring forth a new argument about which games are more cloud-friendly than the others. Considering all the pros and cons of cloud gaming, any gamer could unquestionably come to the conclusion that cloud gaming is the best solution that would enable a user to get high quality graphics that are usually delivered by NVIDIA graphics cards, made available in a regular low-cost smartphone that has the minimum ability to connect to the internet and play HD video.

Cloud Gaming Platform

In this section we discuss about a software that operates in a similar manner to remote desktops and video on-demand services[18] in which the gaming applications are stored and executed on a virtual machine provided by a public cloud, and streamed as a video output through a client software installed in the user's device that could be either a physical computer or a smartphone. The player inputs are handled by the client and are sent back to a server running in the virtual machine that is executing the game during the entire session.

Cloud gaming platforms eliminate the need to purchase expensive niche hardware and installation of games that might consumme several gigabytes of data in case of heavyweight games that require superior system configuration. It supports a diverse range of playable devices including mobile devices like smartphones, tablets, desktop computers, laptops or even digital media players. Besides requiring a high-speed and reliable internet connection for delivering low-latency video streams to the client software, cloud gaming platforms curtails the cost of playing heavyweight games to a huge margin and makes way for new innovative pricing models that shift the traditional distribution via retail outlets to the public cloud providers provisioning the virtual machines symbolizing the only resource for computational power capable of processing complex gaming applications. Involvement of virtual machines as the primary computational resource puts forth the option of choosing the Operating System (OS) in which the games would be played. There are several games that are playable only on Windows OS and it is impossible to play these games on any other OS unless the game production company makes an official release supported by other OS. The most exciting fact about cloud gaming platform is that it eradicates this OS dependency and enables the user to play a Windows supported game in an

Android[20] smartphone that is technically impossible but improvised to run on any device that is capable of running HD video.

The below Figure 2 shows the design and working of a typical cloud gaming platform including the relationship between every component that runs independently to complete the data flow that completes the process of playing a game over a reliable network connection.

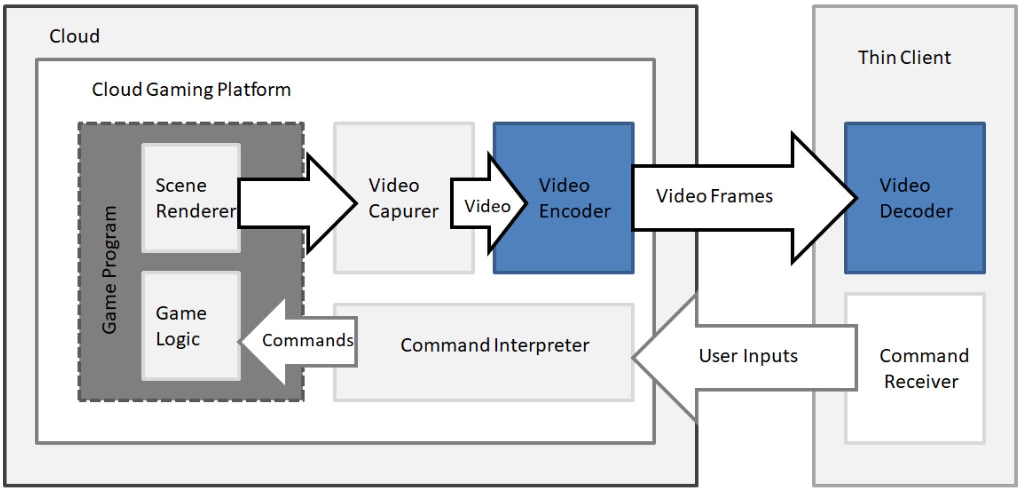

Figure 2: Cloud Gaming Platform Architecture

As seen in the above diagram, a cloud gaming platform consists of two programs - Gaming Server and Gaming Client. While the server program is hosted and run on a virtual computer provisioned by a public cloud provider, the client is a lightweight application that is installed in the player's device (phone, tablet, computer) and receives the video feed containing the game output that is transmitted by the server through the network. The gaming application is usually an executable file that runs on the virtual machine and uses it's CPU to process the gaming data and user instructions, whereas the server uses a *video capturer* sub-program to capture the frames coming in the video output of the gaming application. This requires a GUI to be pre-installed in the virtual machine that is explained in detail in section 3. The captured video frames are then sent to the *video encoder* that is another sub-program that is responsible for compressing the video using x264[19] library that basically reduces the video size without affecting its detail and quality to an extent that is acceptable as an HD video. This is referred to the Live video stream mentioned in the previous section while comparing cloud gaming platforms with live video broadcasting, that is transmitted over the internet and received by the *video decoder* sub-program running inside the client application present in the device being used by the end user who reacts to this video representing the actual game events and sends the reaction via *command receiver* that records the timestamp of the event and the action specified by the user. These commands are mostly coded messages or API calls that are sent back to the *command interpreter* present inside the server program and in turn, interfaces with the gaming application and provides the user commands to the *game logic*.

There are several cloud gaming platforms that provide Gaming as a Service to users who do not want to go into the requirements and complexities behind running a game on a physical computer. TechJury[16], a well-known team of software experts who thoroughly test and subscribe to all aspects of software in order to create impartial reviews, has objectively ranked and contrasted nine best cloud gaming service providers in the year 2021: -

1. Vortex[21]
2. Boosteroid[22]
3. Shadow[23]
4. Amazon Luna[24]
5. GeForce NOW[25]
6. Playkey[26]
7. Paperspace[27]
8. Playhatch[28]
9. GameStream[29]

The above cloud gaming platforms are proprietary and closed-source, thereby making it extremely difficult to use them for experimental and research purposes. Therefore, we have adopted GamingAnywhere[30], the first open-source cloud gaming platform that has disrupted the gaming industry due to its features, services, characteristics and its stupendous configurability, having attracted several computer scientists and developers to perform intensive research work and proven to be better in multiple aspects than many other proprietary cloud gaming platforms.

Gaming Anywhere

GamingAnywhere is the first open-source cloud gaming platform that has been designed for high extensibility, reconfigurability and portability. It supports both Windows and Linux operating systems and is also portable to other operating systems including MacOSX and Android. In previous research work [13], [14], [15] it has been found that GamingAnywhere has achieved not only provides superior video quality but it also achieves high responsiveness while keeping network traffic to a minimum unlike other proprietary cloud platforms that tend to generate heavy network traffic[31]. Furthermore, game developers, service providers, researchers and gamers are able to customize GamingAnywhere with their own configuration in order to fit their requirements. This openness is the prime value that places GamingAnywhere is a different position of prestige and status unlike the rest of proprietary cloud gamimg platforms like On-Live and Gaikai that is closed source and cannot be expected to provide the same openness. Being the first open-source cloud gaming platform, GamingAnywhere is still in its infancy[30] as research projects are still going on regarding its performance and quality that focuses on the value delivered to the end user. This makes GamingAnywhere the ideal tool that serves a key purpose in this project where it is adopted for its openness and integrated with a front-end web application that adds more value for the end users and provides greater meaning and completeness to a cloud gaming system that unambiguously provides Game as a Service.

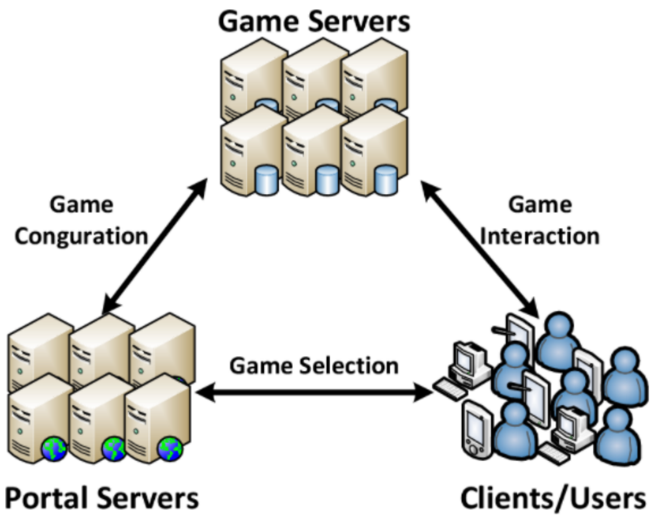

Figure 3: GamingAnywhere deployment scenario[13]

The above Figure 3 shows the deployment scenario of GamingAnywhere. The users are supposed to first login via portal servers that provides a catalog containing the list of available games. A user can select the preferred game and sends a request to play that particular game (GA Server). Upon receipt of this request the portal server has to find an available game server and launches that selected game on that available server while returning the server URL back to the user. This URL is provided to the client application (GA Client) and the client starts receiving the live video feed of the game environment allowing the user to start playing. This portal server is the component on which this research paper is based upon, i.e., a web-application that interfaces with public cloud and orchestrates appropriate cloud resources in order to deliver Game as a Service to the end user while taking care of the management and monitoring of these cloud resources.

The above Figure 4 shows two types of network flows - *data flow* and *control flow*. The data flow is responsible for streaming audio and video (A/V) from the server to the client while the control flow is being used in the reverse direction in order to send the user's actions coming in response to the game events, from the client to the server. This architecture of GamingAnywhere allows it to support a diverse range of games that typically run on PC as well as web based consoles.

GamingAnywhere has two main components: -

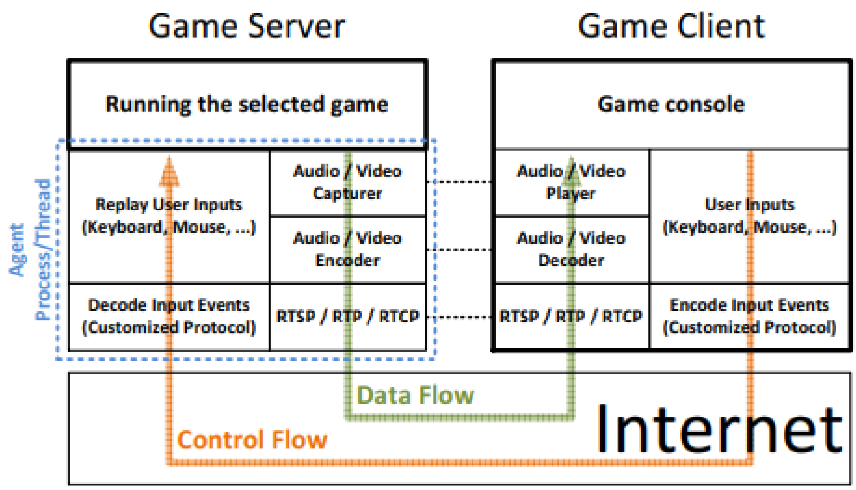

Figure 4: GamingAnywhere architecture[13]

1. **GA Server** - This is a specialized application containing different software modules implemented in autonomous threads that are generalized and similar to those shown in Figure 2. This is the software component that is responsible for capturing the video output of the games that are running on the game server (cloud virtual machine). Along with the selected game, there is also an agent running on the game server that can either be a stand-alone process or a software module injected into the running game depending on the type of game and how it is implemented in the server. Firstly, the agent captures the A/V frames of the game and encodes them using x264[19] codec that is a default value in the initial configuration which can be changed according to the requirement of the user, and then deliver these encoded A/V frames to the client via the data flow. Secondly, the agent interacts with the game and plays the game by reacting to the game events immediately after receiving the user's actions (interrupts from keyboard / mouse / joystick / gesture events) from the client, behaving as a real user itself by imitating the users actions like following orders in a master-slave relationship.

2. **GA Client** - The client is a software that can be said as a heavily customized game console implemented by combining a multimedia player and a keyboard/mouse logger (touchscreen in case of Android smartphones). It is a remote desktop client that is responsible for displaying the real-time game screen that are being captured by the GA Server and delivered in the form of encoded A/V frames. Unlike the GA Server there are only two worker threads that are dedicated to rendering the received A/V frames (coming via data flow) and handling the user inputs (going via control flow) respectively. The below Figure 5 shows the screenshots from the GA Client application for Android smartphones. Before starting to play a game, a user needs to create a *profile* that defines the set of necessary configurations needed to receive the A/V

frames from the respective GA Server. In Section 3 we have created virtual machines that have specific IP addresses. Here is where the specific IP address of a specific virtual machine that is running a game and streaming via GA Server, needs to be set as a mandatory value for creating a profile. Additionally, Controller configuration has been provided that allows a user to disable the controller in case any user decides to spectate a live game being played by another user, without disturbing the gameplay. Audio configuration has been provided for the situations when user needs to downgrade the audio sample and reduce the audio quality in order to adapt to a slow internet connection.

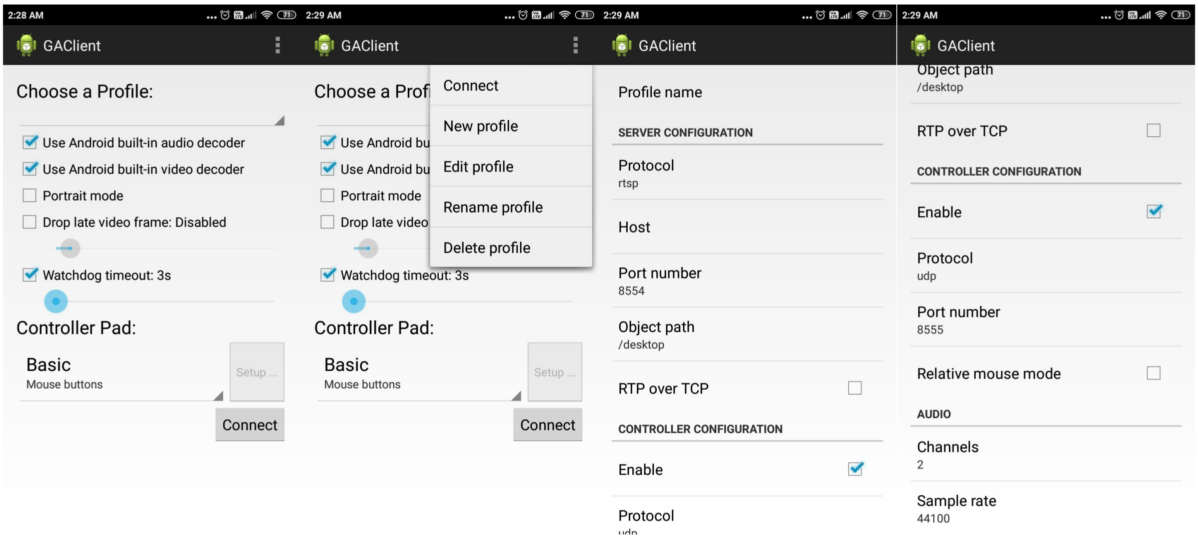

Figure 5: GamingAnywhere Client for Android[13]

3 Deployment and Management of cloud games using AWS

In this section we setup the GamingAnywhere cloud gaming platform within our virtual resources acquired from one of the most popular public cloud provider - Amazon Web Services. Once the setup is complete and the resources are ready, it is integrated with our middleware application playing the role of a portal server in the deployment scenario of GamingAnywhere using the API Gateway provided by AWS so that the resources could be managed and monitored by the end user via the customized components of our application.

3.1 Setting up virtual resources

Before acquiring the virtual resources it is essential to create an account in Amazon Web Services. Therefore, we have created a free-tier AWS account[38] using a valid email and phone number for identity verification including an active credit card that is mandatory for billing while the user needs to accept these statutory terms and conditions[35] in order to continue with accessing the creation of necessary virtual resources required to perform all the experiments related to this research project. More such details about user access provisioning in AWS can be found in the official website[38]. This is compulsory as using the SSH client we would run commands and shell scripts in the CLI that is a part of the setup activity in which we install the necessary softwares and update the libraries required to run the gaming application. Firstly, we need to install a GUI within the newly created EC2 instances as a GUI is the minimum requirement for a user to be able to view the screen or the video output of a particular virtual machine.

3.1.1 Acquiring virtual machines

Once the AWS account is ready for use, the first thing we do is to create virtual machines, i.e., EC2 Instances[39]. This is the most important component of the experiment as it is the virtual computer responsible for running the gaming application and also the GamingAnywhere server simultaneously. Before launching and connecting to an Amazon EC2 instance, AWS IAM[40] need to be accessed in order to generate a key-pair[41] as the security credentials for accessing the EC2 Instances. Creating these EC2 instances require access to the AWS Main Console[39] in which one can capitalize on the highly engaging and user friendly user interface that provides a step-wise guide to complete the creation of a virtual machine. We have chosen the default and most recommended operating system - Amazon Linux 2[47] as our default operating system since our experimental gaming applications are extremely light in terms of graphics and performance. In this case, we would be creating t2 micro instances[43], the most popular instances widely available in the free-tier plan and having a high quotient of availability over all regions[44]. The configuration of t2 micro instances is very basic with a single core CPU, 1 Gigabyte of RAM that are enough to run a basic gaming application that does not demand huge resources for Graphics and performance, and are considered as ideal virtual machines for research and experimentation. During the creation of an EC2 instance, we also need to create and attach a virtual storage in order to provide the file transfer and storage capability to the virtual machine. AWS provides EFS[45] that has an incredibly high file transfer rate along with the option of scaling up according to the storage resource demands of the application. It is highly recommended that we follow the necessary steps[42] to establish a SSH connection to the newly created EC2 instances in order to update the necessary libraries, drivers and software repositories immediately after the EC2 instances start running.

3.1.2 Installing a Graphic User Interface

Once the EC2 instances are created and in 'running' state, we connect with an instance via SSH using PuTTY - SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers[56]. The below Figure 6 shows a screenshot of the PuTTY console that is connected to a newly created EC2 instance.

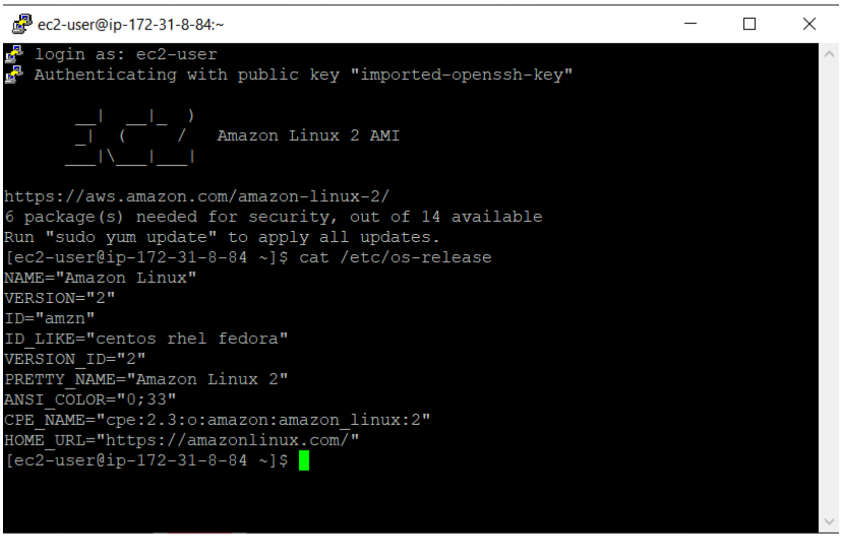

Figure 6: PuTTY Client

Following the official documentation[57] that helps us install the MATE[58] desktop environment that is a lightweight GUI based on GNOME[59] 2 available for Amazon Linux 2. This is followed by the installation of the Virtual Network Computing or simply the VNC service. TigerVNC is one of the most popular client applications that is being used to bridge the gap between a virtual computer and a real user. It is a high-performance, platform-neutral implementation of VNC, a server/client application allowing users to launch and interact with graphical applications on virtual machines. It provides every level of performance that is necessary to run video and three-dimensional applications, and it attempts to maintain a common appearance and allows re-use of components[60]. Installation takes several minutes until all the necessary libraries are installed correctly, The below Figure 7 shows the screenshot of the completion stage of VNC server installation within the EC2 instance.

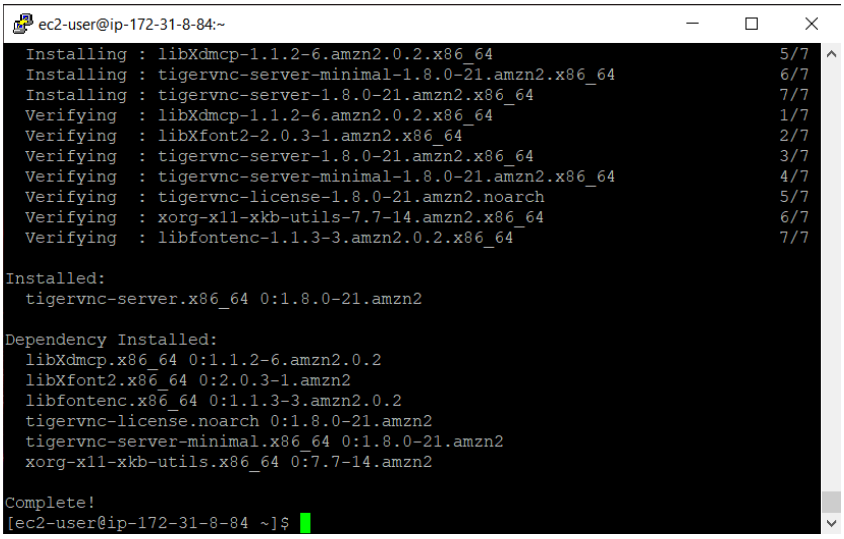

Figure 7: VNC Server Installation

3.1.3 Installing GamingAnywhere

In this sub-section we perform the most important task of installing and configuring GamingAnywhere - an open source Cloud Gaming Platform mentioned in the earlier sections. We download the entire repository[61] from GitHub and place it in the virtual storage of the EC2 Instance. We follow the GamingAnywhere official quick start guide[62] to build and resolve the dependencies of GamingAnywhere by first editing the **env-setup** script and setting the absolute path to **gaminganywhere/deps.posix** directory and then merging the environment variables using the **source** command. This is followed by the building of this repository by navigating to **deps.src** directory and running the **make** command, followed by the ultimate build by running the **make all** command in the **ga** directory. Once the building of dependencies is complete then we can proceed with the installation step by running the **make install** command in the same **ga** directory and observe that the generated files have been installed into the **bin** directory. Once the building of GA Server is done, we need to find suitable gaming applications that are relatively lightweight in terms of resource consumption. Therefore, we download the sample games and configurations from GamingAnywhere downloads[63] page where there are plenty of portable free games that are suitable for testing with the GamingAnywhere system. In this project we have selected the following games for experimentation: -

1. Armagetron Advanced[64]

2. Assault Cube[65]

3. Neverball[66]
4. Sauerbraten[67]
5. LEGO Batman[68]

While these games are extremely lightweight, portable and completely free to deploy, LEGO Batman is a proprietary commercial game that is not always available for testing and only the configurations are made available as open-source material. Moreover, this concludes the initial setup of virtual machines that serve the purpose of providing the prime resource of computational power that is responsible for running two different applications simultaneously - GA Server and an executable packaged application, i.e., the actual game which is supposed to be played by the end user. Performing the above execution of commands bring out significant changes in the existing file systems present in a newly created EC2 instance.

3.1.4 Provisioning Storage

According to the default system configuration of a standard t2 micro instance category, each instance has been equipped with an Elastic Block Storage (EBS)[70] of 8 Gb when the first EC2 instance was created. After creating four more EC2 instances, The default 8 Gb EBS of each instance automatically downscales to adjust the total EBS quota limited to 30 Gb as per the free-tier account allowances. An EBS attached to a particular EC2 instance persists the data throughout the lifetime of the instance thereby appearing as the secondary storage of a virtual machine. However, the storage space that comes with EBS is loosely coupled with the EC2 instance and the storage space is allocated from a separate pool of memory resources owned by the Cloud Provider. In order to provide a tightly coupled storage with an EC2 instance, AWS offers instance store[69] that can be used as the backing storage[71] for an Amazon Machine Image (AMI)[72] that in turn comes out as a packaged virtual computer with tightly coupled storage block acting as its hard disk. This would be a rational way to imitate a physical computer with a powerful CPU with an incredibly fast SSD as its secondary storage device, by building a similar system using virtual resources available in Public Cloud that could save its entire state inside a larger memory block, in this case, Amazon S3[73] bucket that is capable of holding data as objects meaning it persists several virtual computers throughout the time specified by the user. Using the above procedure to create gaming instances could theoretically prove to produce virtual computers that are more efficient in terms of storage and file transfer. More discussion about gaming using custom AMIs have been discussed in the later sections. Furthermore, the gaming applications selected for this research project are extremely lightweight and demand very low processing speed and runtime memory. Considering these factors, every EC2 instance has been configured and dedicated to playing one particular game only. The below Figure 8 shows a screenshot of the AWS console having the newly created EC2 instances, that are named according to the gaming application that is present in its memory. The properties of every EC2 instance, such as instance name, instance id, instance state, public IP address etc., canbe seen within the AWS console itself, and would be imported and displayed in the external web application through Cloud API gateway that is discussed in the next section. Although the gaming executable applications require very less amount of space, installation of GamingAnywhere, VNC Server, MATE desktop and GNOME together amount to an approximately extra 300 Mb of space in the storage of the virtual machine. Since these extra files and folders are a compulsory requirement for the cloud gaming platform to run seamlessly, it is advised from a developers' perspective to retain the EBS blocks in the modified state so that this configured EC2 Instance could be used as a source instance image that is subject to replication whenever a user requests a new virtual machine for a new game, as running the above commands and installation of the required files are time consuming and cannot be considered as a convenient operation using an automated script every time a new EC2 instance is created, as this time consumption would significantly increase the startup delay and affect the user tolerance in a negative way.

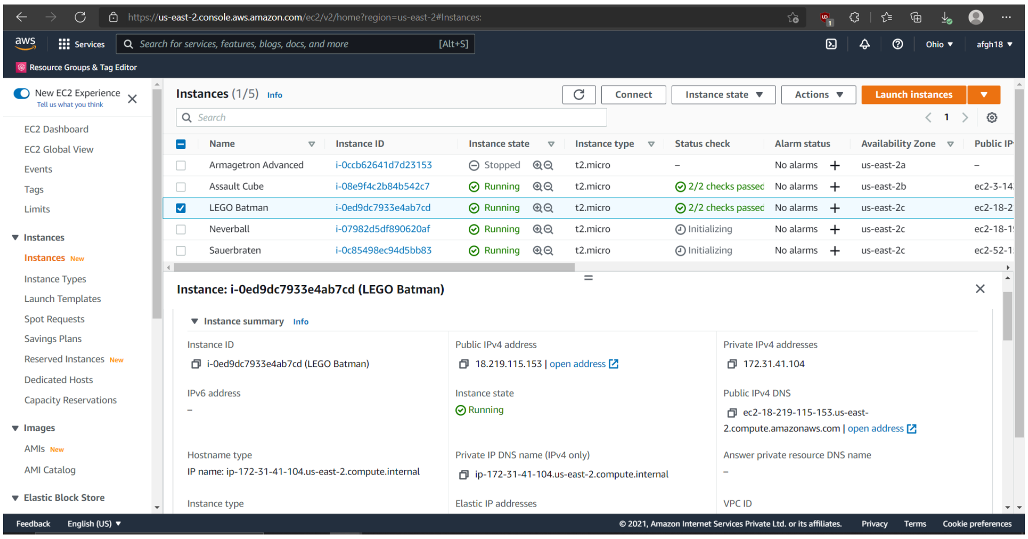

Figure 8: AWS Console with EC2 instances

3.2 Interfacing with Public Cloud

In this section we arrive at the main focus area of this research project that involves the actual development of a web application and its integration with the Public Cloud where we have previously created and configured our virtual resources and kept them in standby for future usage in the relevant experiments. In previous research works, scientists have analyzed the future scope of Cloud Gaming Platforms within a specific pool of target users[74]. Unlike the most popular cloud gaming platforms like On-Live and Gaikai that are proprietary and closed-source, a newly emerging cloud gaming service provider needs to develop a front-end application for gamers that would help them choose their desired games from a well-defined catalog that could be decorated using the newest web-designing libraries like Bootstrap[75] and made to look futuristic enough to attract thousands of gamers and hobby players who are looking for cheaper and more convenient ways of playing games on lightweight devices without worrying about the resource requirements. When compared to the immense time and effort required to develop a cloud gaming platform from scratch, adoption of an open source cloud gaming, i.e., GamingAnywhere curtails not just the the time and development-cost, but also makes a prodigious reduction in the time that is needed to launch a complete cloud gaming service into the market. The below Figure 9 shows the integration of a web application with the public cloud that is having three main components, i.e., API Gateway, Serverless Function (Lambda)[76] and the EC2 instances that we had created in the previous section. While we would go into detail about these necessary components in following sub-sections, our prime development work has been done in building a web application that acts as a bridge between the user and a cloud gaming platform. In this case, the web application interfaces between the user and public cloud by giving access to the user to operations and management of the virtual resources and also orchestrates the provisioning of EC2 resources according to the user demand.

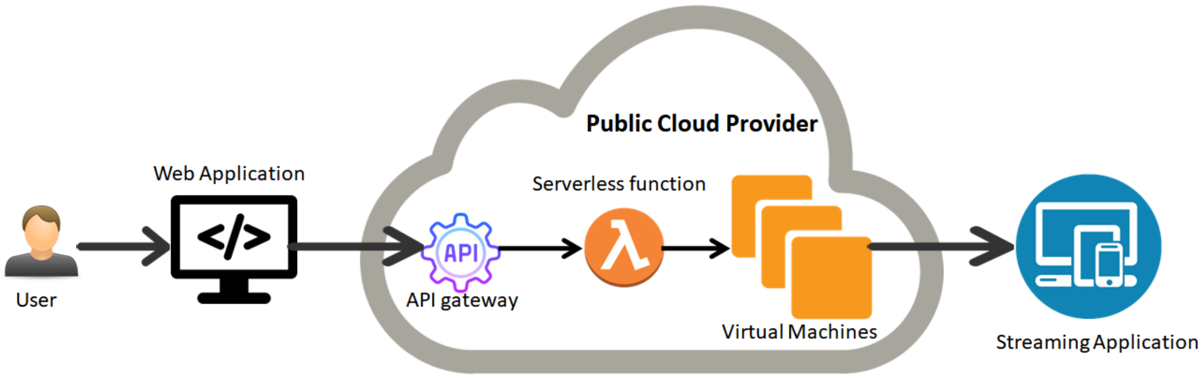

Figure 9: Integration with Public Cloud

3.2.1 Web Application Components

As discussed earlier, this web application is a bridge between the user and the cloud computers and the purpose of building such an application is to provide a user friendly interface that provides controlled access the users to handle the cloud resources closely while imitating the relationship between a regular user and a physical computer. According to the research done in[74], this web application completes the system design of a proper cloud gaming platform while providing gaming services to the users as well as encapsulating and abstracting the underlying complexity of the operations happening in the background, i.e, the process in which a user is able to start playing a particular game that is installed in a dedicated EC2 instance. The front end of this web application has been developed using react.js[77] - A JavaScript library for building user interfaces. The reason for using this specific library is due to the ability to create interactive user interfaces and designing simple views for each state in the application while it also enables the efficient update and rendering the right components triggered by a change in data. React enables the developer to build customized components that are able to encapsulate and manage their own state. Since all the code is written in JavaScript[78], it allows the developer to avoid using templates and pass rich data without rewriting existing code. Using this web application, the user could easily view the number of gaming instances available under a definite public cloud. The components of this web application are able to communicate with the public cloud through an API gateway that enables a user to manage these resources. Moreover, in a user's point of view, the existence of the main mechanism of running a game is completely virtual while the front-end application is the only face of the whole cloud platform that is responsible for interacting with the user and delivering the real value for which a user chooses to consume the services offered by a cloud gaming platform. Keeping these requirements in mind the following components have been developed in order to impart the statutory functionalities to the end user: -

1. *User Sign up* - This is the functionality that allows a new user to create a new account using a valid email id and choose a new password as security credentials. In order to provide better user management, persistent database[91] has been added in the next section that is capable of holding and micro managing user data efficiently in a cloud database[92] service.

2. *User Login* - This is the functionality provided to the existing user having a valid user account to gain access to the main page of the web application after the authentication is successful. Basic functionalities of user access provisioning have also been added which helps users to reset password using the forgot password option and recover existing user account. Inclusion of external authorization API calls such as Google login would make it easier for the users to login quickly without filling up forms for registering. The objective is to deliver more value by engaging the users on a social networking level by including more such third party authorization from social networking website servers such as Discord and Facebook.

3. *User Logout* - This is the functionality that enables a user to terminate a session after which the user would be needing to provide the credentials once again to gain access to the main page.

4. *Play Button* - This is a button provided within the card component that is in binding with an API call to the AWS API Gateway in order to invoke a lambda function that is capable of switching on (changing the state) an EC2 instance that runs an automated script required to initiate the GA Server and a specific gaming application. More such controls can be provided to the end user in form of buttons that could be used to indirectly invoke lambda functions that can shut down or terminate an EC2 instance. As an attempt to make the UI more user friendly, programming logic has been added internally inside the component that changes the button color according to the value of instance state retrieved via cloud API.

5. *VM Card* - This is the customized component that is specially designed to represent a specific EC2 instance whose properties like instance name, instance id, instance state, etc., are populated within the fields, overall giving it a card-like appearance that is prominently distinguishable for the user to understand the details of that particular virtual machine. This is the most important component that provides the main functionalities for controlling the EC2 instances through the buttons that are mapped indirectly to lambda functions via cloud API calls.

6. *VM List* - This is the main page of the web application that is responsible for displaying the available virtual machines i.e., all the cards representing all the available virtual machines in one page that allows the user to get an overall picture of the resources and the details altogether in a collective view using the flexbox[79] component that is robust, configurable and automatically adjustable according to the number of items getting displayed in this flexbox.

The below Figure 10 shows a screenshot of the main page of this web application that shows all the available EC2 instances that were created and configured in the previous section within the specified AWS free-tier account.

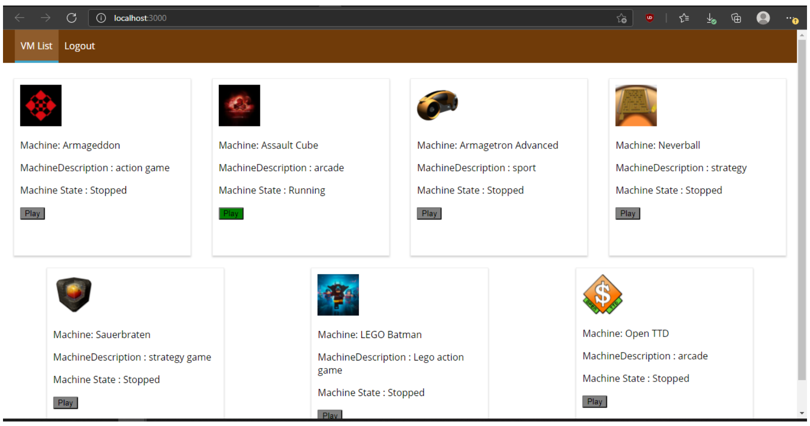

Figure 10: Main page of the web application showing available EC2 instances named according to the game that is installed in each of them

In the previous section, we had created these virtual machines using the AWS console and it is possible to view all these properties of each EC2 instance in the AWS console itself. However, in this case all these data are being derived from public cloud as a result of an API call that returns a JSON payload containing all the data about the available EC2 instances present within the specified AWS free-tier account. This is happening due to the integration of the web application with a serverless Lambda[76] function that is programmed only for this purpose of providing the required data to the web application via AWS API gateway[80] discussed in the next section.

3.2.2 Cloud Components

In this section we discuss about the components that are provided by the public cloud and are utilized by the user in order to complete the integration with the web application. Although the cloud space is public, the resources are heavily protected with several layers of security, hence making it quite difficult for external web applications to access the virtual machines. Therefore, it is necessary to use a combination of such such cloud services in order to fulfill the requirement of operations and management over the virtual machines. Figure 6 shows the integration of the web application with the public cloud in which we use a combination of API gateway and serverless Lambda function in order to establish a connection between the web application and EC2 instances.

1. **API Gateway** - This is an interface that links all cloud services together allowing one program to make its functionality and data available to external programs and connect diverse software components to interconnect across a network. AWS provides REST API that connect via HTTP protocol and perform operations like create, read, update and delete from a specific data pool. In this project we have created a REST API that exposes multiple endpoints that are used inside the web application to make HTTP calls such as GET, POST and DELETE in order to retrieve the available EC2 instances, switch on or off the instances and terminate existing EC2 instances.

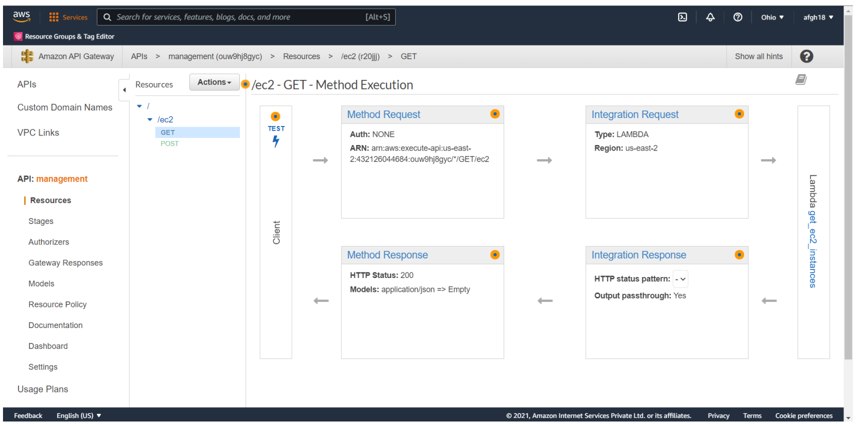

Figure 11: AWS API Gateway

The above Figure 11 shows the console view of an AWS API gateway that has an endpoint accessible through HTTP GET and it is mapped to a serverless Lambda function that is supposed to retrieve the requested values and pass to the API in form of a JSON response body.

2. **Lambda function** - This is a serverless[81] programmatic function dedicated for a single purpose. Being serverless, it allows developers to build and run applications without deploying into separate servers that are managed by AWS. It supports code belonging to various languages such as JAVA, Python, Node.js etc., by providing a runtime API for executing code without provisioning a

server for running the program. In this case we have used Python to program these Lambda functions in which a dedicated library for EC2, **boto3**[82], is the official Python SDK for AWS and has been used to access the specified free-tier AWS account via dedicated IAM role created solely for granting full EC2 access to the Lambda function. We have created multiple Lambda functions for multiple purposes and mapped them to multiple API endpoints that serve as trigger for that particular Lambda function.

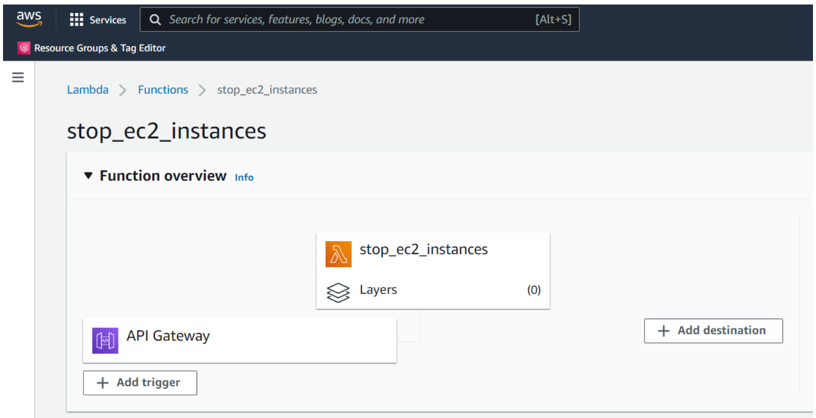

Figure 12: Lambda function mapping

The above Figure 12 shows a Lambda function created for a single purpose, i.e., to stop an EC2 instance or multiple instances, once the mapped API endpoint is externally invoked and the Lambda function is triggered automatically.

It is imperative to understand that AWS Lambda plays a key role in this system design as Lambda functions are the only single-purposed microservices that perform the actual operations based on the type of requests sent by the user through the web application. Therefore, it is necessary to create multiple Lambda functions in order to represent the different types of actions that are required to be performed upon the cloud resources.

The above Figure 13 shows the screenshot from AWS Console having the python programming logic using boto3 library, that is written inside a lambda function dedicated to fetch the available EC2 instances. Upon execution of this program, the lambda function returns a JSON response containing the data about available EC2 instances. The *instance id* is a necessary value that need to be displayed in the front-end components of the web application along with the *instance state* so that user can understand the situation of the cloud computational resources. The *instance id* is also required to be passed into separate programming logic within the web application in case of switching on / off or terminating a specific EC2 instance having a specific *instance id*. This is a part of managing the resources as discussed in the next sub-section. According to the official documentation[83] of boto3, it is possible to derive multiple values about EC2 instances such as the ip addresses of the EC2 instances. As discussed in section 2 within GA Client, this ip address need to be supplied to

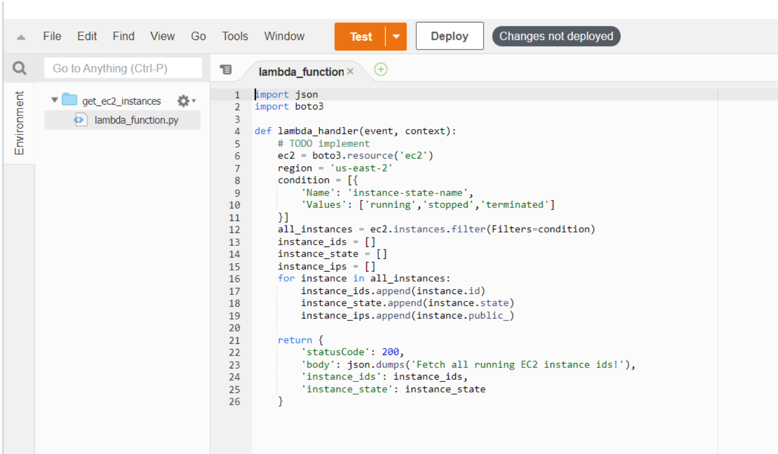

Figure 13: Lambda function to fetch available EC2 instances

the GA Client while creating a profile for the user to play a game. However, it is not recommended to extract multiple values of EC@ instance properties, as it creates a significant upsurge in the complexity of the programming logic and results in the increment of execution time of a lambda function. The effect of this practice has been discussed in the evaluation section where our primary focus is to optimize the system design such that the overall startup time of this service stays as low as possible.

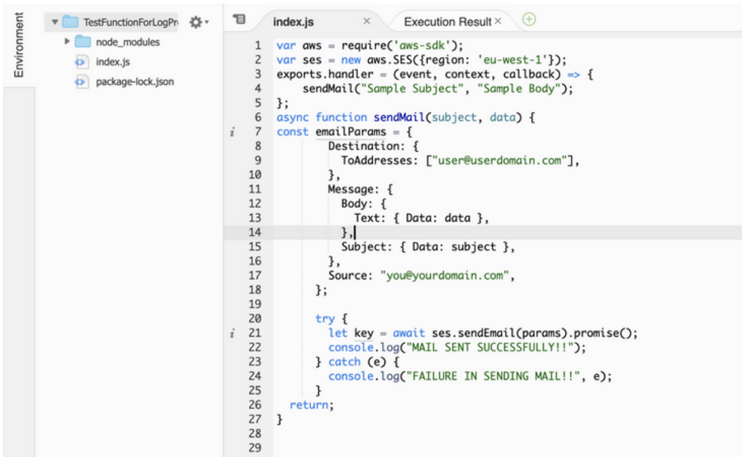

Figure 14: Lambda function to send e-mail

The above Figure 14 shows the screenshot from AWS Console having the javascript programming logic written inside a lambda function dedicated to send out an email to a user having a verified email id. The purpose of this lambda function is a part of monitoring of cloud resources discussed in the next sub-section where a separate service is built to send out notifications at regular intervals decided by the developer in order to create a reporting service. Keeping in mind the pricing scheme of AWS on Lambda functions[76], these functions are executed separately by the cloud servers managed by the cloud provider, creating a dependency on the duration of execution for each lambda function. Therefore, it is vitally important to decide the number of lambda functions to be created as it affects the overall costs and expenses required to develop this project.

3.3 Management of resources

The main purpose of the web application is to be a bridge between the users and cloud gaming servers as suggested by [13] that a web portal is necessary to interface and orchestrate with the public cloud. Furthermore, the web application should be capable enough to perform the management operations according to the user demands and monitoring operations independently as programmed by the developer.

In this section we take the system design to the next level by improving the architecture by adding more software components to the initial design. According to the best practices in software development, it is highly recommended to keep a loose coupling between the front-end application and the back-end API service. Previously, we had designed the system keeping a tight coupling between the front-end application and the AWS API Gateway. The below Figure 15 shows the high-level overview of an improved system design that consists of a middle layer between the front-end web application and the public cloud.

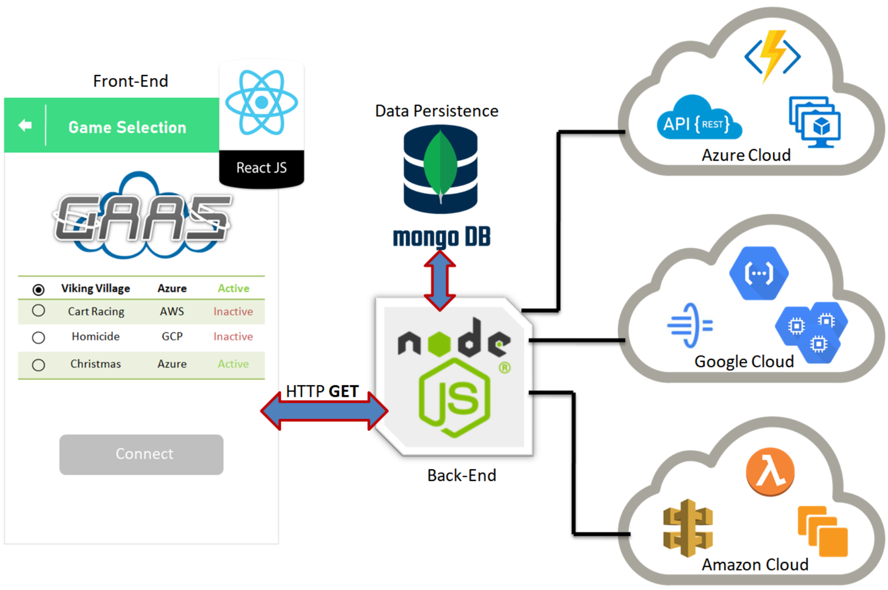

Figure 15: System design for managing resources

The front-end application has been decoupled from the AWS API gateway and instead, it fires API calls to a stand-alone API service built using Node[93] javascript. There is also a Mongo[91] database added to the system that plays the role of a persistent database while holding the data about available EC2 instances. The back-end API service has callback[94] functions that are mapped to the route event that occurs when the front-end application fires an API call to a specific endpoint mapped to a certain callback function that fires the relevant API call to AWS API Gateway in order to complete the request that technically appears as a flow of nested events. Once the AWS API Gateway triggers the relevant lambda function, the request is complete and the necessary data is returned to the front-end application that starts reflecting the change. Moreover, when the front-end application fires an API call in order to fetch the data about available EC2 instances, the callback function present in the back-end application fires another API call that executes the lambda function and returns a JSON response body. The same callback function then receives the JSON response, parses the data and pushes it to the persistent database such that the next time when user requests to fetch EC2 data, the back-end API service would simply retrieve the existing values from the persistent database instead of executing the lambda function again and again. Hence, this practice has reduced the number of unnecessary executions of lambda functions and has unequivocally reduced the overall cost to a huge margin.

As seen in Figure 15, the back-end web application is efficiently designed to create multiple combinations of API routes, callback functions and cloud API calls in order to interface with multiple cloud providers including Microsoft Azure[84] and Google Cloud Platform[86]. Unlike AWS, Azure and GCP both have different methods to retrieve VM information like *instance id*, *instance state*, *availability region* etc. In both the cases, the cloud provider has exposed existing API endpoints that can be consumed to fetch the data about virtual machines without executing a serverless function that even eliminates the need for creating a dedicated serverless function for retrieving the available virtual machines. Using the Azure REST APIs for accessing virtual machines can be found in [85] and the similar method for GCP can be found in [87]. Using different public cloud providers for management of compute resources brings out different values in the time consumed for the completion of requests affecting the overall QoS that is further discussed in evaluation section.

3.4 Monitoring of Resources

Cloud resources are based on the pay-as-you-use[90] model that requires usage to be properly monitored and recorded in logs as the usage decides the costs charged by cloud provider. Every cloud provider has software services that are dedicated only to monitoring of other resources using a notification service that implements a message queue to post auto generated messages at regular intervals set by the developer. Moreover, monitoring ensures the consistent efficiency in resource utilization and holds a mandatory role to play in building a cost-effective usage strategy. Cloud providers like AWS, Azure and GCP have their own versions of monitoring services that come by default as a supervising service over compute services and is responsible for raising alarms and send email notifications to the user, based on the default usage quota set according to the free-tier usage allowances by the cloud provider. Although there are different monitoring plans and strategies for different cloud providers, the common thing here is that every monitoring scheme is structurally based on the publisher-subscriber or Pub/Sub model as said in [89] for AWS cloud and [88] in Google cloud. Therefore, one could propose multiple ways of building a custom monitoring system that would publish messages to a message queue or a topic and then send notifications triggered by specific events depicting changes in the cloud environment, to multiple subscribers that are listening to that topic.

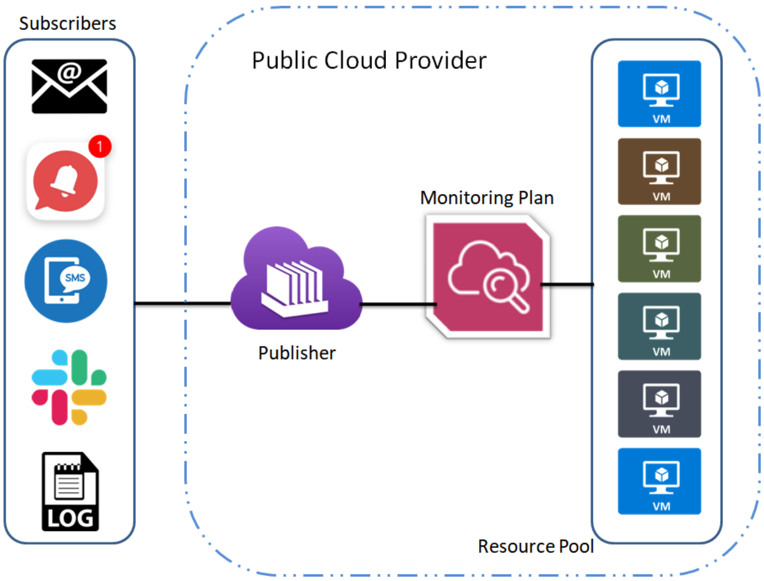

Figure 16: System design for monitoring resources

The above Figure 16 shows the system design of a basic strategy for monitoring the virtual machines used for running the games and the cloud gaming server. Every notification is based on a cloud event that denotes a change in the cloud environment. One can create a custom cloud event or select what to do about the existing cloud events like high cpu utilization, low memory available, storage quota exhausted, etc. Every cloud event can be mapped to trigger a notification that would publish a message in the queue so that subscribers can get a stream of event notifications. There can be multiple systems for sending notifications to a subscriber depending on the combination of different subscriber, time interval, cloud events and resource pool.

In the previous sub-section we have discussed about a lambda function in Figure 14, that is designed only for the purpose of sending an email notification to a user with a valid email address. This particular lambda function is being used by a cloud monitoring plan defined in the service CloudWatch[51] provided by AWS Cloud as a highly reliable and configurable monitoring service that allows the developer to set custom events to either publish notifications via AWS Simple Notification Service (SNS)[54] or trigger a lambda function that is capable of creating notifications and send to multiple subscribers.

The above Figure 17 shows a lambda function sending out email notifications via AWS Simple Email Service[55] (SES), a cost-effective, scalable and flexible email service enabling developers to send an email from within a running application. In section 3.2.2 we have discussed the concerns over executing multiple multiple lambda functions altogether that raises the cost of consuming these services and sending out emails every now and then would undoubtedly increase the overall execution time of

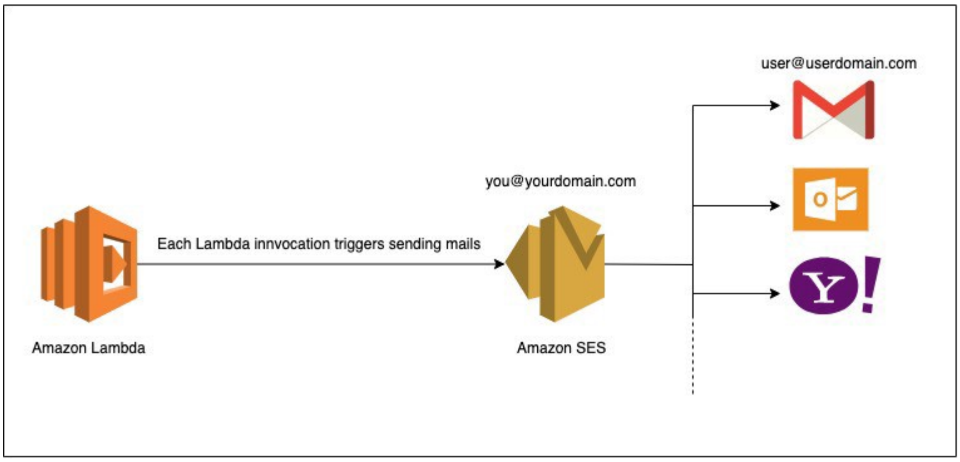

Figure 17: Lambda function for sending out notifications

lambda functions.

The below Figure 18 shows a more improved and advanced structure of a custom monitoring service that is based on the Pub/Sub model using Amazon SNS, a fully managed service for messaging that supports both application-to-application (A2A) and application-to-person (A2P) communication. This pub/sub model provides messaging topics delivers high throughput with its push-based many-to-many messaging between applications and microservices.

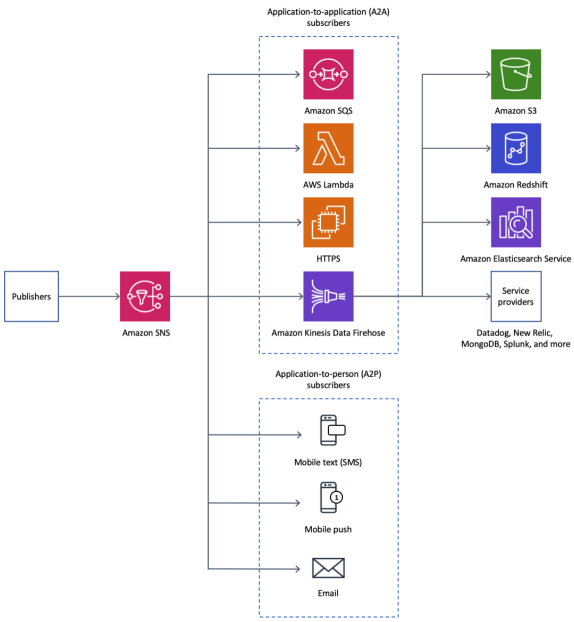

Figure 18: Complete monitoring service using AWS SNS

It is recommended to use the pub/sub model for sending frequent messages at regular intervals for monitoring EC2 instances. Unlike the previous approach where a

CloudWatch event triggers a lambda function, the publisher notifications in SNS are sent by CloudWatch events instead of invoking a lambda function. While the A2A communication involves messaging between independentent software components like distributed systems, event-driven serverless applications, lambda functions and HTTP endpoints, the A2P functionality sanctions the sending of messages directly to a user via text messages, push notifications in internet enabled smartphones and emails.

According to [50], in order to use CloudWatch to monitor EC2 instances, the developer needs to first define the events that would serve as benchmarks for the overall monitoring plan. In this case, both the user and the administrator needs to be notified about the running EC2 instances at regular short intervals referred to as *heartbeat messages*. In order to implement this functionality, the developer needs to decide the interval between two messages and create a rule that defines a recurring cloud event that occurs periodically after that specified time interval. Creating an event rule requires the developer to set a *rate expression* [53] that starts when a scheduled event rule is created and then continues working independently. Rate expressions are written in the **rate(value unit)** format where *value* is a positive number and *unit* is the unit of time that can be hours, minutes, days in case value is greater than 1 and hour, minute, day in case value is equal to 1. We have used AWS Command Line Interface[48] (CLI) to implement a new CloudWatch event by creating a rule with the help of **put-rule** command[49] as shown below: -

```
aws events put-rule
--schedule-expression "rate(1 minute)"
--name HeartBeatRule
```

This particular event rule satisfies our requirement for provisioning a system that fires an event every 1 minute and the name of this event is set as *HeartBeatRule*. According to the monitoring system design in Figure 18. This new *HeartBeatRule* is used as a trigger for SNS notifications that continuously publishes messages that can be consumed in multiple ways by multiple subscribers and displayed to the user.

4 Evaluation and Optimization

In this section we evaluate the performance of the entire cloud gaming system consisting of GamingAnywhere platform integrated with the middleware web application in order to deliver a seamless gameplay along with an engaging user friendly user interface that provides value and more meaning to this project. We would focus on the most important factor of performance evaluation, i.e., time consumed to start the service or simply startup-time or startup-delay. In order to come to a judgement, we need to analyze this amalgamated scenario by breaking it into smaller phases. The time measurement typically starts from the point when user presses a button to start the gaming server, i.e, the time taken for a request to process from the web application to reach the AWS API Gateway (T1), Then the AWS API Gateway redirects this request to a lambda function that starts the EC2 instance containing the selected game. Ergo, the time taken for the lambda function to execute (T2). Finally, the time taken for the EC2 instance to change its state from *stopped* to *running* is the largest time interval experienced by the user (T3). Following this, the GA Server and the game starts automatically as the EC2 instance starts using an automated shell script and the A/V frames start transmitting the video output of the welcome screen of that particular game (T4). In the final phase, one needs to manually enter the IP address of the game server while creating a profile in the GA Client android application, as discussed in Section 2. Currently there is no way to automate the startup of GA Client and populate the configuration values in the creation of a new profile. Hence, we would not be considering this time as it is a completely manual approach and might show inconsistent readings in time duration depending on the user's personal speed in setting up the GA Client. Therefore, the total startup time (T) can be calculated as: -

$$T = T1 + T2 + T3 + T4 \tag{1}$$

According to [13] the time taken for GA Server to start is called the *Response Delay* (RD) that is the sum of *Processing Delay*, *Playout Delay* and *Network Delay*. In the experiments done in [14] it has been found that the *Processing Delay* and the *Playout Delay* for the Batman[68] game is 27 milliseconds and 12 milliseconds respectively. Whereas, *Network Delay* is a variable that depends on network conditions and can change values from 5 to 20 seconds depending on the network conditions. Even with an incredibly fast internet connection, it is possible to face significantly large *Network Delay* due to traffic congestion within the network. The time consumed for an EC2 instance to change its state from *stopped* to *running* is about 33 seconds for an instance that was created and configured initially. Another test case was performed when this prepared instance having GA Server and the game installed in it along with the dependencies, is used as a source AMI[72], and a request is fired to create a new EC2 instance from the custom AMI that we have built in Section 3. This process takes much longer than switching on an existing EC2 instance, as the recorded time for this process has been observed to a maximum of 38 seconds. Hence, this difference in starting up a game server is observed due to the difference in the background procedure that allocates an EC2 instance to serve the user requirements.

Coming to the execution time of lambda functions, it has been recorded that lambda functions may take up to 15 minutes[46]. However, this is depending on the code complexity and size of the programming logic written inside a lambda function. In this case, the programming logic is extremely short and compact, that results in an execution time ranging in few milliseconds.

As discussed in the beginning of this paper, the requirement of bandwidth for Cloud based games are significantly larger when compared to that of traditional online games. When we talk about the differences in characteristics, there are two perspectives of making this observation: -

1. between various cloud gaming platforms

2. between various games played on the same platform

Considering that all the popular Cloud Gaming platforms are owned by multi billion dollar companies, it is safe to assume every gaming platform has been backed up by state of the art cloud infrastructure, capable of delivering the best performance to the end user in any deployment scenario. Although a lot of research work has been done to compare the difference in transmission characteristics between different cloud gaming platforms, such as Manzano et al. [32] collect and compare network traffic traces of OnLive and Gaikai, including packet inter-arrival times, packet size, and packet inter-departure time, to observe the difference between cloud gaming and traditional online gaming from the perspectives of network load and traffic characteristics.

Differences between various games in the same platform can be very high, upto several Mbits, significant impacts on game performance and QoE of end user can be witnessed.

Shea et al. [7] measure the interaction delay and image quality of OnLive system, under diverse games, computers, and network configurations.

Provisioning the most convenient platform for cloud game developers is still a big challenge mainly because of the complexity, heterogeneity and distributed nature of the cloud gaming platforms. There is a barter between optimization room and development complexity. According to [8] there are two kinds of cloud gaming platforms - transparent platforms and non-transparent platforms. Transparent platforms opt for either very low or even no supplementary development complexity and run unmodified games having very limited scope for optimization. Whereas in a non-transparent platform, more optimized performance is delivered in exchange of supplementary development complexity such as recompilation of code and augmentation. Deploying new games on a transparent cloud platform has been made easier in exchange of a suboptimal performance. Windows remote desktop protocol is one of the oldest protocols used for remote connection with a server that has been used to run the RemoteFX extension allowing Windows servers to leverage Hyper-V virtual machines and GPUs to run several remote applications as well as cloud games. The experiments done in [9] shows that Windows servers automatically adapt to network dynamics while RemoteFX extension is running the show. However, the QoE reduces significantly due to subservient responsiveness and an unusually high rate of frame loss. Moreover, [10] has demonstrated another cloud gaming platform having both service platform and rendering system as independently distributed subsystems along with an encoding and streaming system to handle the output, supporting multiple clients, browser based clients and isolated audio and video capturing.

Both Depasquale et al. [9] and Kim et al.[10] are closed-source and proprietary platforms, making them less suitable for cloud gaming research. Whereas, GamingAnywhere [33], [34] is the first open-source transparent cloud gaming platform. Its design boasts about advantages like extensive, portable, configurable, and openness. It is shown in [13], [14] that GamingAnywhere outperforms several commercial, closed-source and proprietary cloud gaming platforms, and is being used and enhanced in several research papers and cloud gaming studies. For example, in Hong et al. [1] researchers have developed adaptation algorithms for multiple gamers, in order to maximize the gaming QoE. In addition to a user study that maps cloud gaming parameters to gaming QoE and also optimizing algorithms for allocation of resources , they have performed major enhancements in GamingAnywhere [33], [34] in order to support on-the-fly adaption of bitrate and frame rate.

Optimization is a vast area of innovation with an infinite scope of possible ways to create combination of configuration values, software components, resource allocation strategies and better process scheduling algorithms. In order to have an efficient optimization strategy, it is imperative to analyze the changes in cloud server architecture, resource utilization patterns, placement of virtual machines and allocation of virtualized hardware accelerators or GPU. Resource sharing in public cloud platforms is also an advantage that contributes in optimization as it enables the automated rapid upscaling of VM configuration without affecting the gameplay whenever the game process demands higher resource allocation than the initial system configuration of the game server, in case multiple players demand to play the same game while other games are not being played, load balancing and autoscaling techniques can be implemented to ensure the provisioning of more resources when demanded by the game. Implementing these optimization strategies reduces the cost of the entire operation. Hence, it is unquestionably a necesary practice that we need to consider while facing changes in Distributed architecture due to vast geographic distribution of clients and servers. This can be mitigated by following one of the advanced concepts of cloud computing, i.e., Edge Computing. Deployment of edge servers would solve the distance issue by being a bridge between server and client and by bringing the services closer to the clients, increasing the overall availability of the cloud servers even in regions having scarce availability of resources. This concept has been scientifically stated and supported in the research paper [36] in which scientists have suggested a distributed architecture that creates and deploys edge servers present very close to the physical location of the requesting player. Another optimization strategy, although more complex and aggressive, is the division of the game engine into various components and distribute these components dynamically on multiple cloud servers based on the demand and availability of resources dedicated to running the games and gaming servers[33].

Last but not the least, the most effective way to serve the purpose of cloud gaming is to attract more and more audience to adopt this new way of gaming, is by delivering higher QoS and QoE. Since the main drawback faced by cloud gaming is network speed, it is crucial to optimize the system such that a low delay tolerance is maintained throughout the life time of the service. However, it is not on the hands of the user to expect a seamless gaming experience without getting affected by network issues. We have to consider network issues as an integral part of the system dependencies that will never cease to affect the end user's QoE including the playability of the game. In this part there comes another factor of the game genre that plays a vital role in deciding the playability and lag tolerance. For example, highly responsive games falling under categories of First Person Shooter (FPS) like Call of Duty, Medal of Honor, HALO, etc., have very low lag tolerance as few milliseconds of lag can cause the player's progress to deteriorate and bringing the overall performance to an extent where user might to quit playing rather than playing with lag. Whereas, strategy games like Age of Empires, Frozen Throne, Command and Conquer demand relatively less responsiveness, thereby allowing the user to tolerate a little bit of lag while enjoying the overall experience without worrying about where the game is consuming computational resources from.

5 Discussion and Future scope

In this project we have studied cloud gaming in detail, starting from the role of cloud computing in the modern gaming industry and the digital disruption caused by the rapid growth of commercial cloud gaming platforms that provide high-quality gaming experience to users, with unimaginable ease of use, cheap pricing and hassle free resource allocation. GamingAnywhere is the first open source cloud gaming platform that has attracted significant amount of attention from a vast community of scientists, researchers, students, business developers and hobby gamers. The immense hard work done behind the research experiments have yielded extraordinary results that prove that GamingAnywhere outperforms most of the commercial and proprietary cloud gaming platforms. This has given rise to development of front-end applications that could adopt GamingAnywhere as the default gaming platform and bring completeness to the architecture, capable of rivalling the existing proprietary cloud gaming platforms as GamingAnywhere has several advantages over other platforms, mainly: -

- It is open-source, responsive, customizable, configurable, flexible, adaptive and OS friendly.
- It generates minimum network traffic unlike On-Live platform that generates traffic even if it is in idle state and is three times slower in terms of network delay[13].
- It has been used heavily in several research work to spark more innovation.
- It opens the field to more designs in software architecture that can engage the target users and deliver them more value.
- It eliminates the need to develop a cloud gaming platform from scratch when integrated with a web application, thereby reducing the time to launch in the market as a complete cloud gaming platform
- It provides security to game developers as the game software does not leave the cloud. Hence, developers do not need to fear software piracy.
- It allows developers to invest more time and effort on improving the product quality without worrying about compatibility

Although AWS has been used as the primary public cloud provider in this research paper, GCP and Azure could also rival AWS services with their equally supreme and capable computational resources. GCP and Azure have one advantage over AWS - GCP and Azure APIs do not require serverless functions to fetch any information related to available virtual machines. This helps in cost-cutting as there is no need to consider the costs of executing serverless functions. Although, GCP and Azure require serverless functions for monitoring purpose, it still reduces the time consumed for resource management as the cloud APIs are capable of handling resource management operations like starting and stopping virtual machines, serving the purpose in a more economic way with minimum number of serverless functions. In the experiments performed in this research paper, only low-end virtual machines with minimum system configurations have been used. However, the future scope demands intensive resource utilization of virtualized hardware acceleration, i.e., provision of a GPU that supports heavyweight games requiring high graphics computation. Furthermore, it can be speculated that this system design is simple and spacious enough to accommodate massive changes in functionalities and addition of more user friendly features and providing promising opportunities to gamers who demand to play heavyweight games on GPU enabled virtual machines with hi-end configuration. Nonetheless, there remains an argument about the costing of retention of virtual resources while they are not in use. Every cloud provider charges money to the user who retains the virtual machines, IP addresses and storage blocks. Ideally, creation of virtual machines on-demand and terminating or destroying them immediately after gaming session gets over, would undoubtedly reduce significant amount of expenses charged for retaining these resources. Besides, this argument can be answered by the classification of games into single player and multiplayer categories. Single player games have a certain game progress that brings about a change in the system memory where the game is installed. Every user expects this progress to be saved in order to continue playing the same game in the next session. Therefore, it is vital and mandatory to retain the virtual machine when it is stopped by the user after the end of a specific gaming session. A good practice would be to use custom AMI that is built and configured by the developer and saved as a snapshot in a large storage service capable of persisting a large data chunk represented by a GPU enabled custom AMI backed by a tight coupled memory block containing game directories and libraries required to run the game. Certainly this would be charged by the cloud provider as the memory buckets where the AMI is saved is priced according to space occupied and the duration for which it is stored. This is cheaper than retaining the original virtual machine and storage block as the account holder only needs to pay for the storage bucket and not for the retention of the virtual machine. Moreover, the front-end application could be enhanced to provide a time-boxed gaming plan for the users that would provide the single player games for a stipulated period of time and then charge extra expenses when the user could not finish the single player game within the time-box and intends to retain the game server in order to play for a prolonged duration. However, this scenario would be different for multiplayer games as there is no concept of game progress in multiplayer games and the players never need to save their progress. Therefore, in this case there is no need to retain the game server, i.e., virtual machines and storage blocks are free to get destroyed immediately after user stops the gaming session and has the provision of getting a game server newly created every time a user decides to start playing a specific multiplayer game.

References

[1] Placing Virtual Machines to Optimize Cloud Gaming Experience Hua-Jun Hong, De-Yu Chen, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu

[2] An Evaluation of QoE in Cloud Gaming Based on Subjective Tests Michael Jarschel, Daniel Schlosser, Sven Scheuring, Tobias Hoßfeld University of Wurzburg, Institute of Computer Science, Germany

[3] K. Chen, P. Huang, and C. Lei, "How sensitive are online gamers to network quality?" Communications of the ACM, vol. 49, no. 11, pp. 34–38, 2006.

[4] Onlive, http://www.onlive.com/

[5] Gaikai, http://www.gaikai.com/

[6] Ubitus, https://ubitus.net/

[7] Cloud Gaming: Architecture and Performance. Ryan Shea and Jiangchuan Liu, Simon Fraser University Edith C.-H. Ngai, Uppsala University Yong Cui, Tsinghua University

[8] A Survey on Cloud Gaming: Future of Computer Games WEI CAI1 , (Member, IEEE), RYAN SHEA2 , (Member, IEEE), CHUN-YING HUANG3 , (Member, IEEE), KUAN-TA CHEN4 , (Senior Member, IEEE), JIANGCHUAN LIU2 , (Senior Member, IEEE), VICTOR C. M. LEUNG1 , (Fellow, IEEE), AND CHENG-HSIN HSU5 , (Senior Member, IEEE)

[9] E. Depasquale et al., "An analytical method of assessment of RemoteFX as a cloud gaming platform," in Proc. IEEE Conf. Medit. Electrotech. Conf. (MELECON), Beirut, Lebanon, Apr. 2014, pp. 127–133.

[10] K. I. Kim, S. Y. Bae, D. C. Lee, C. S. Cho, H. J. Lee, and K. C. Lee, "Cloud-based gaming service platform supporting multiple devices," ETRI J., vol. 35, no. 6, pp. 960–968, Dec. 2013.

[11] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, ChunYing Huang, and Chin-Laung Lei, "Measuring the latency of cloud gaming systems," in Proceedings of ACM Multimedia, Nov 2011.

[12] Esa Piri, Matti Hirvonen, and Jukka-Pekka Laulajainen, "Empirical Evaluation of Streamed Online Gaming over WiMAX," in Testbeds and Research Infrastructure. Development of Networks and Communities, pp. 255–270. Springer, 201

[13] Chun-Ying Huang, Kuan-Ta Chen, De-Yu Chen, Hwai Jsung Hsu, and Cheng-Hsin Hsu, "GamingAnywhere The First Open Source Cloud Gaming System," ACM Transactions on Multimedia Computing, Communications and Applications, vol. 2, no. 3, pp. 1–26, 2010

[14] Chun-Ying Huang, De-Yu Chen, Cheng-Hsin Hsu, and Kuan-Ta Chen, "GamingAnywhere: An Open-Source Cloud Gaming Testbed," in Proceedings of ACM Multimedia 2013 (Open Source Software Competition Track), Oct 2013

[15] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. "GamingAnywhere: An Open Cloud Gaming System"

[16] Keelan Balderson, "9 Best Cloud Gaming Service Providers for 2021". December 3, 2021. https://techjury.net/best/cloud-gaming-service/

[17] WiMAX Wikipedia Page. [**?**]

[18] Enderle, Rob (2019-10-11). "Is xCloud – the Xbox game streaming service – a glimpse at the future of the desktop?". Computerworld. Retrieved 2020-06-04. https://www.computerworld.com/article/3445366/is-xcloud-a-glimpse-at-the-future-of-the-desktop.html

[19] Video LAN Official website - x264. https://www.videolan.org/developers/x264.html

[20] Android Operating System Official Website. https://www.android.com/intl/en_in/

[21] Vortex Official Website. https://vortex.gg/?aff=13258514

[22] Boosteroid Official Website. https://cloud.boosteroid.com/auth/register?utm_source=b2b&utm_medium=manual&utm_campaign=7YLSY&utm_content=&subid=21jranu137ff

[23] Shadow Official Website. https://shadow.tech/

[24] Amazon Luna Official Website. https://www.amazon.com/luna/landing-page/?ie=UTF8&linkCode=ll2&tag=techjury02-20&linkId=8437ab48b0f6b6e6e2b3cc2efbbeb446&language=en_US

[25] NVIDIA GeForce NOW Official Website. https://www.nvidia.com/en-eu/geforce-now/

[26] Playkey Official Website. https://playkey.io/en/

[27] Paperspace Official Website. https://gaming.paperspace.com/

[28] Playhatch Official Website. https://www.playhatch.com/

[29] GameStream Official Website. https://www.gamestream.biz/

[30] Gaming Anywhere - An Open Source Cloud Gaming Platform. https://gaminganywhere.org/

[31] TOWARDS UNDERSTANDING THE RELATIONSHIP BETWEEN GAME TYPE AND NETWORK TRAFFIC FOR CLOUD GAMING Mirko Suznjevic1, Justus Beyer2, Lea Skorin-Kapov1, Sebastian Moller2, Nikola Sorsa1

[32] M. Manzano, J. A. Hernandez, M. Uruena, and E. Calle, "An empirical study of cloud gaming," in Proc. 11th IEEE Annu. Workshop Netw. Syst. Support Games (NetGames), Venice, Italy, Nov. 2012, pp. 1–2.

[33] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," ACM Trans. Multimedia Comput., Commun., Appl., vol. 10, no. 1s, pp. 10:1–10:25, Jan. 2014.

[34] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere: An open cloud gaming system," in Proc. ACM Multimedia Syst. Conf. (MMSys), Oslo, Norway, Feb. 2013, pp. 36–47

[35] H.-J. Hong, C.-F. Hsu, T.-H. Tsai, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Enabling adaptive cloud gaming in an open-source cloud gaming platform," IEEE Trans. Circuits Syst. Video Technol., vol. 25, no. 12, pp. 2078–2091, Dec. 2015.

[36] T. Kämäräinen, M. Siekkinen, Y. Xiao and A. Ylä-Jääski, "Towards pervasive and mobile gaming with distributed cloud infrastructure," 2014 13th Annual Workshop on Network and Systems Support for Games, 2014, pp. 1-6, doi: 10.1109/NetGames.2014.7008957.

[37] AWS Service Terms and Conditions. https://aws.amazon.com/service-terms/

[38] AWS Free Tier - https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields

[39] AWS Elastic Computing - EC2 - https://aws.amazon.com/ec2/

[40] Amazon Web Services - Identity and Access Management. https://aws.amazon.com/iam/

[41] Security Key-Pair - Amazon Web Services - Linux EC@ Instances. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

[42] Amazon Web Services - Connect to EC2 Instances. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-methods.html

[43] Amazon Web Services - EC2 T2 Instances. https://aws.amazon.com/ec2/instance-types/t2/

[44] Amazon Web Services - Regions and Availability Zones. https://aws.amazon.com/about-aws/global-infrastructure/regions/

[45] Amazon Web Services - Elastic File System. https://aws.amazon.com/efs/

[46] Amazon Web Services - AWS Lambda enables functions that can run up to 15 minutes. https://aws.amazon.com/about-aws/whats-new/2018/10/aws-lambda-supports-functions-that-can-run-up-to-15-minutes/

[47] Amazon Web Services - Amazon Linux 2.https://aws.amazon.com/amazon-linux-2/

[48] Amazon Web Services - AWS Command Line Interface. https://aws.amazon.com/cli/

[49] Amazon Web Services - AWS CLI Reference - put-rule command. https://docs.aws.amazon.com/cli/latest/reference/events/put-rule.html

[50] Amazon Web Services - CloudWatch - Schedule Expressions for Rules https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

[51] Amazon Web Services - CloudWatch Events. https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html

[52] Amazon Web Services - EventBridge. https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html

[53] Amazon Web Services - Cron and Rate Expressions. https://docs.aws.amazon.com/lambda/latest/dg/services-cloudwatchevents-expressions.html

[54] Amazon Web Services - Simple Notification Service. https://aws.amazon.com/sns/

[55] Amazon Web Services - Simple Email Service. https://aws.amazon.com/ses/

[56] Putty Official Website. https://www.putty.org/

[57] Amazon Web Services - GUI on Amazon EC2 running Amazon Linux 2. https://aws.amazon.com/premiumsupport/knowledge-center/ec2-linux-2-install-gui/

[58] MATE Desktop Environment Official Website. https://mate-desktop.org/

[59] GNOME Official Website. https://www.gnome.org/

[60] Virtual Network Computing - TigerVNC Official Website. https://tigervnc.org/

[61] GamingAnywhere Source Code Repository on GitHub. https://github.com/chunying/gaminganywhere

[62] GamingAnywhere Quick Start Guide. https://gaminganywhere.org/doc/quick_start.html

[63] GamingAnywhere Downloads Page - Sample Games and Configurations. https://gaminganywhere.org/download.html

[64] Armagetron Advanced. https://gaminganywhere.org/dl/games/ArmagetronAdvancedPortable_0.2.8.3.1.paf.exe

[65] Assault Cube. https://gaminganywhere.org/dl/games/AssaultCube_v1.1.0.4.tar.bz2

[66] Neverball. https://gaminganywhere.org/dl/games/NeverballPortable_1.5.4_Rev_2.paf.exe

[67] Sauerbraten. https://gaminganywhere.org/dl/games/sauerbraten_2013_02_03_collect_edition_linux.tar.bz2

[68] LEGO Batman. http://games.kidswb.com/official-site/lego-batman/2

[69] Amazon Web Services - Amazon EC2 instance store. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html

[70] Amazon Web Services - Elastic Block Storage. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html

[71] Amazon Web Services - AMI Types. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ComponentsAMIs.html#storage-for-the-root-device

[72] Amazon Web Services - Amazon Machine Images (AMI) . https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

[73] Amazon Web Services - Amazon Simple Storage Service. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#CoreConcepts

[74] Chun-Ying Huang, De-Yu Chen, Cheng-Hsin Hsu, and Kuan-Ta Chen. GamingAnywhere: An Open-Source Cloud Gaming Testbed

[75] Bootstrap Official Website - Build fast, responsive sites with Bootstrap https://getbootstrap.com/

[76] Amazon Web Services - AWS Lambda. https://aws.amazon.com/lambda/

[77] React JavaScript Official Website.https://reactjs.org/

[78] JavaScript Official Website.https://www.javascript.com/

[79] React Native - Layout with Flexbox https://reactnative.dev/docs/flexbox

[80] Amazon Web Services - AWS API Gateway.https://aws.amazon.com/api-gateway/

[81] Amazon Web Services - Serverless Computing. https://aws.amazon.com/serverless/

[82] Amazon Web Services - Boto3 Python Library. https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html

[83] Boto3 Official Documentation. https://buildmedia.readthedocs.org/media/pdf/boto/latest/boto.pdf

[84] Microsoft Azure Official Website. https://azure.microsoft.com/en-in/

[85] Microsoft Azure - Virtual Machines - Get. https://docs.microsoft.com/en-us/rest/api/compute/virtual-machines/get

[86] Google Cloud Platform Official Website. https://cloud.google.com/

[87] Google Cloud Platform - Method: instances.get. https://cloud.google.com/compute/docs/reference/rest/v1/instances/get

[88] Google Cloud Platform - What is Pub/Sub? https://cloud.google.com/pubsub/docs/overview

[89] Amazon Web Services - Pub/Sub Messaging https://aws.amazon.com/pub-sub-messaging/

[90] Pay-as-you-use - Cloud Computing. https://en.wikipedia.org/wiki/Pay-as-you-use

[91] Mongodb Official Website. https://www.mongodb.com/

[92] Mongodb Atlas - Multi Cloud Database. https://www.mongodb.com/atlas/database

[93] Node.js Official Website. https://nodejs.org/en/about/

[94] What are callbacks? https://nodejs.org/en/knowledge/getting-started/control-flow/what-are-callbacks/

[95] Team Viewer Official Website. https://www.teamviewer.com/en-us/

[96] LogMeIn Official Website. https://www.logmein.com/

[97] Microsoft Remote Desktop. https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-allow-access

[98] How Sony bought, and squandered, the future of gaming. Sean Hollister. Dec 5, 2019, 8:00am EST. https://www.theverge.com/2019/12/5/20993828/sony-playstation-now-cloud-gaming-gaikai-onlive-google-stadia-25th-anniversary