Study on Game Streaming in Web Browser

***Abstract*— This paper deals with the study of different possible approaches to implement a cloud based gaming platform on AWS. There are different possible approaches to implement this service using multiple service architectures. This paper talks about some of such approaches. To cope up with the increasing resource needs for different computation problems we recommend using cloud as a platform perfectly fit for it. This paper talks about various challenges faced when AWS is used to deploy such gaming service platform. Finally, it gives a clear idea about implementation of a Steam-based gaming server on a VM using Microsoft Azure.**

I. INTRODUCTION

Cloud plays a huge role in the daily activities these days. Enterprises are looking to move their resources onto cloud. Cloud offers companies to scale their infrastructure to fit their needs accordingly from time to time. As cloud services always use the most innovative technology, cloud users are bound to have good performance and will have a competitive edge over others. Cloud technologies also enable to get applications to market quickly, without worrying about underlying infrastructure costs or maintenance. All of these have led to enterprises to shift their services onto the cloud. This has also led to the development of online cloud-based game streaming. A lot of applications are in the development stages focusing on deploying low latency high performance online game streaming.

II. DESCRIPTION

A. Problem to be solved

In this project, we would like to develop a possible solution to develop an online distributed game streaming application that can be used from anywhere using a web browser/application. This application would allow the user to play any game present on the cloud without having to download it onto the local machine. The game would essentially be running on the cloud service the video and audio from it would be relayed to the user. At the same time, the gaming controls or actions that the user would input would be sent over to the game running on the cloud service. All of these processes must be completed with low latency to provide a good experience for the user.

B. Tools and frameworks that has been used

We are implementing this project with these tools but as per the requirements in near future they might vary as the project proceeds towards completion.
Amazon AWS, EC2, API's, Microsoft Azure (WebRTC, Mousepointer, Gamepad, Parsec, Web Audio,), javascript, Nodejs etc.,

1. Networking – WebRTC

With WebRTC, you can add real-time communication capabilities to your application that works on top of an open standard. It supports video, voice, and generic data to be sent between peers, allowing developers to build powerful voice- and video-communication solutions. WebRTC helps us in communicating flawless with the server to produce low latency results. This plays a key role to design a game service.

2. Video – Media source extensions

Playing video and audio has been available in web applications without plugins for a few years now, but the basic features offered have really only been useful for playing single whole tracks. With Media Source Extensions (MSE), this is changing. MSE allows us to replace the usual single track src value fed to media elements with a reference to a MediaSource object, which is a container for information like the ready state of the media for being played, and references to multiple SourceBuffer objects that represent the different chunks of media that make up the entire stream. MSE gives us finer grained control over how much and how often content is fetched. As games contain a lot of data which represent the display that change with every action of the user, MSE and the usage of MediaSource object enables us to manipulate them in a more convenient manner.

3. Audio – Web audio API

Web Audio API describes high level Web API for processing and synthesizing audio in web applications. It helps us create an overall audio rendering using multiple AudioNode objects which are interconnected. The introduction of the audio element in HTML5 is very important, allowing for basic streaming audio playback. This is still not good enough for sophisticated games and other interactive applications. Also, Web Audio API can be integrated conveniently with WebRTC API which is very much needed in any game as sound latency should also be avoided.

4. Input/Output messages

The input prompts performed by the user while playing the game are passed to the server from the browser's suspects which are taken as commands like top (keyup), down (keydown) etc., and if the user wants to play the game using a game-controller we are trying to implement it using the one of the GAMEPAD API's available. There is a considerable difference in how the commands are activated. The keyboard and mouse commands are initiated using events whereas the controller button prompts are activated in the form of polls. Usually there can be lags when the cursor is used. So, in order to circumvent these errors, we use **Pointer Lock API** which accurately reflects and passes the information to the server over time.

5. Methodology

We are trying to implement this web application where the user is made to visit a designated website which contains the gaming application. Through which the users are directly connected to the AWS server where the game is selected, and the prompts given by the player are accepted and delivered to the server in AWS. The processing for the game is done in

the server by the GPU and the components set for the cloud. The audio, video, config signals are preprocessed which go through encodings and decoding and sent as output to the player playing the game. This project can be integrated with the parsec API and AMI for real-time implementation or it can be built from scratch where the gaming platform can be created but with both the models have limitations such as the size of the game, GPU processing power, availability of bandwidth and other resources. All this is for the successful and seamless playing experience for the player.

III. IMPLEMENTATION

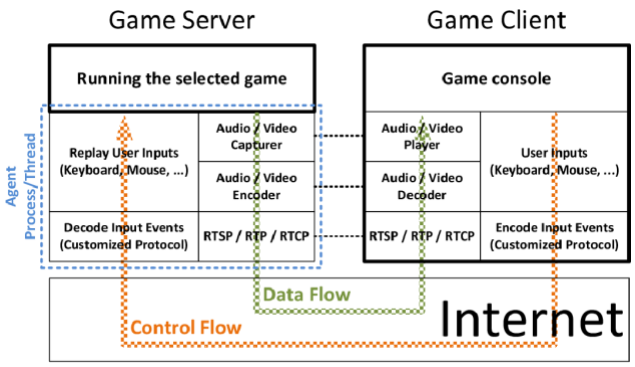

Fig 1: Structure of the gaming platform

The possible methodologies we feel suited to implement this service and challenges faced during the implementation are described as follows:

First, we need to create the server-side, client-side javascripts to make sure to connect the requests and responses between the client's web-browser and VM which acts as a real-time server.
In order to run the games, we have taken a Game-boy emulator to load and run the .nes collection of games.
Need Nodejs for running the javascripts.

All the methodologies are an attempt to implement this platform on AWS.

Method-1:

Combination of Services:
VM(EC2 instance) + The scripts + Emulator + Games files

Problems:
Now that you have everything set-up, there comes the real problem for this implementation.

1. How do you connect the application to the a link where you can deploy the project ?

   Let's say you can open a port on the VM to run the script. How do you connect the User's system to the with a link that could direct the user's browser to the localhost of the VM ? A connection issue.

2. How to generate multiple instances ?

   It could work for this project but how about 5 players need to run AAA titles they all can't share the same resources. They need multiple instances of the same VM to be running.

3. Screencast of the VM's application to the user's browser can be done but found it hard to integrate interactive support from user's browser to the app in VM through WebRTC.

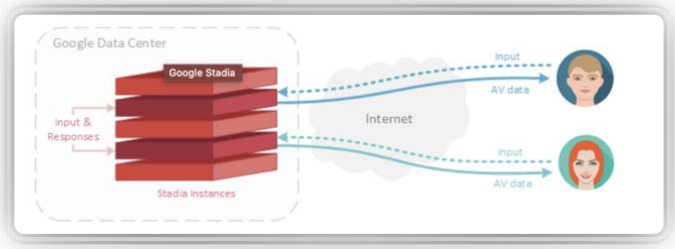

Fig 2: Creation of instances for every new user

To solve these kinds of problems we had to shift to a different approach with different service architecture.

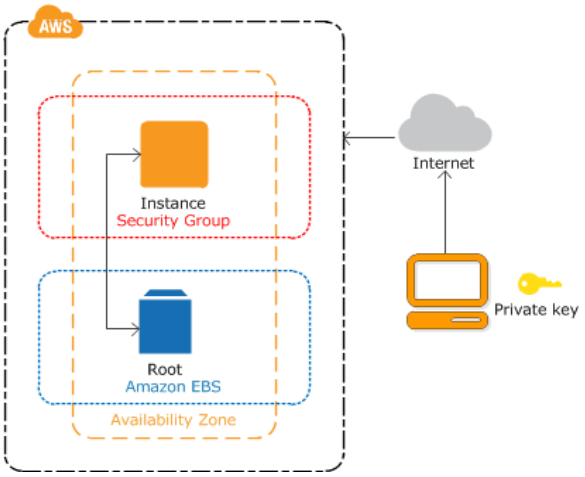

Fig 3: Model - 1

Method – 2:

S3 + CloudFront + Lambda triggers:

Let's place all our scripts as a in an S3 bucket. Now go to the Amazon CloudFront service where it creates a portal to access files and deploy them onto the cloud with a proxy website link. For the deployment we will be connecting the scripts to the user's browser using lambda triggers, they help in the communication of viewer's request and response with server script's request and response (called origin request and response) with the CoudFront playing the managerial role to manage this content-based network.

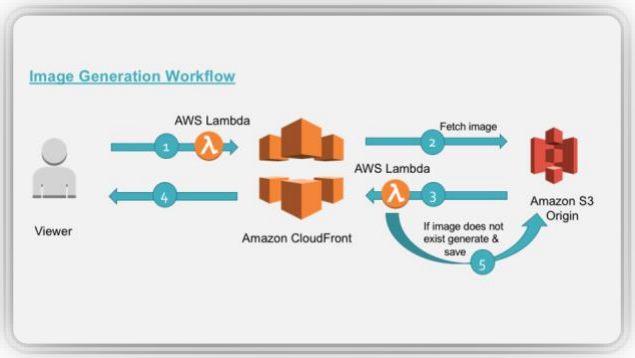

Fig 4: Model - 2

Problems:

- The Emulator is in the form of .exe, so it can't be accessed with a lambda function.
- What if we want to access the emulator with it's API, then it needs some commands to be run on a powershell/cmd which the Lambda function doesn't provide as it doesn't have a VM on it.

But, this method might possibly be a potential solution to our problem, as many CDS (Content delivery services) like Amazon prime, Gaming platforms like King use this kind of architecture for their operations.

Method-3:

VM + lambda + CloudWatch + CloudFront

This solution might actually solve the problem but as we can access the files in the VM with Lambda Function and multiple instances can be initiatied with CloudWatch, CloudFront acts as a proxy by giving us the link to direct users to the VM.

Fig 5: Model - 3

But no such combination exists where we can access files in VM with lambda function.

So, we have opted a model to demonstrate that it can still be implemented as many companies like Google, Nvidia, Steam, Parsec etc., are actually implementing such cloud gaming platforms in real time but has a different network, service and programming architecture that works perfectly.

Steam Model:

For this model we are using the VM from Microsoft Azure where we set the preferences as the following:

- Windows 10 2016 DataCenter
- NV4as_v4 (4 vcpus, 14 GiB memory)

Here we had to opt for NV4 series system instead of NV3, these systems have AMD Raedon Instinct M125 which is not suited for gaming. So, if you have NV3 opt it as it is equipped with a Nvidia M60 graphic card which is much suitable for gaming.

Set the security groups for this project to enavle all RDP, ICMP, UDP protocols

Change the security, Computer management, auto defragmenting drives, auto-login system policies and configurations before installation of visual and audio drivers on the VM.

Install the appropriate Graphic and sound drivers in the VM. For this VM we used the following drivers:
Video - AMD Raedon Instinct M125
Audio - VB-Cable

Now we need to configure a VPN such that both the VM and the user's machine are interconnected. This can be done with the help of zero-tier VPN where we could configure the VPC between necessary devices.

Disable IPv6 in VM as some softwares try to do a IPv6-over-IPv4 tunnel which would actually create a connection problem between devices.

We also need to adjust the MTU on the VM to optimize network traffic and size of the packet to steam remote-play.

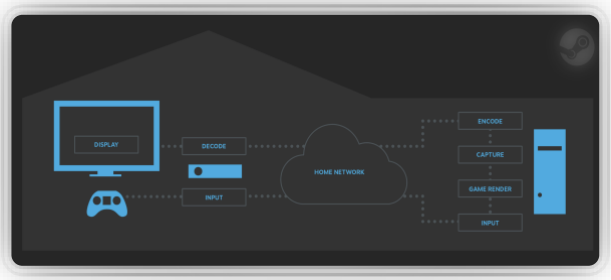

Fig 6: Steam model for this project

This is the overall structure of our network now. Here we were trying to build the right side of the Home network as shown in the figure above from scratch at the beginning phase of our project but now it has already been configured by steam for us.

Now that everything is read for us, install steam in the VM and login with your account. Make sure to set your steam to be on beta version, as it supports remote-play option. Change

the other settings like prioritizing network traffic, hardware encoding, adjusting performance of graphic cards on steam etc.,

Now install and login to steam on your local machine, make sure to join the VM's network through zero-tier VPN, Enable remote-play and adjust settings such that the performance on VM is maxed out and the priority is given to client machine from the local device by changing advanced settings for client i.e., VM. Also, change the limit of bandwidth in steam settings to 30 Mbits/sec on local machine as unlimited isn't working.

Now, try installing the game you like on the VM's steam. When done, if every setting is perfectly configured on VM and local machine you will be able to get an option as shown below.

Fig 7: Option of stream on Steam with interconnected machines

Steam it by selecting the VM and run the games you want. You will get a screen which looks similar to the figure below.

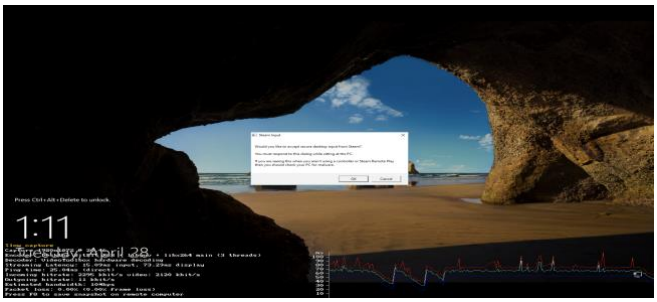

Fig 8: Screen indicating success of game streaming

Fig 9: Statistics of working module

Look the remote-play streaming statistics where:

- The ping time is 71.26 ms (in this case for me)
- Red line – decoding time and latency.
- Light blue – data transfer to client side
- Dark blue – encoding the video in H264 format on server side.

The lower the ping rate the smooth the gameplay. If you get larger ping times, try changing the VM location-based ping times to different server locations.

There might be issues because remote play is still on a beta version.

The same kind of procedure can be used on EC2 instances in AWS as well but we had some problem installing drivers on instances on VM's in AWS.

But there is the problem of security as we would be turning off windows firewall on VM to allow the un-interrupted gameplay.

IV. What we learnt

Through this project we gained an idea of how to implement a real time streaming service for games. We possibly had an idea of how to map the cloud service architecture from a 20,000 ft view to a 5000 ft view. We learnt about implementing WebRTC between devices like screen casting, message passing etc., we gain the knowledge about how to approach to the solutions by designing different possible architectural solutions to the problem. All the hardware drivers, policies also need to be modified and brought to a compatible medium such that they work collaboratively to provide the experience.

V. What furthur future advances can be made

This paper deals with the possible solutions to the problem. They might be implemented with resources and services that are actually described in the models. We have taken the possible solutions based on some sources and but i believe they can be implemented with different set of programming modules. We have seen the possible solutions of this problem with the integration of Go + nodejs + API + network management but it was hard to decode and learn given the time constraint. It's a lengthy and challenging project but can be implemented. We had to use a sample code developed by others to deploy it in our instance and models but they didn't work due to its constrained, new but extensive working schema.

VI. ROLES

This project was all about researching on working models and experimenting on various cloud services. We did our parts learning about different programming languages to use, packages to be used, drivers to be selected, what to be selected for the better integration and results.

VII. REFERENCES

[1] *https://ieeexplore.ieee.org/abstract/document/7536162*
[2] *https://onlinelibrary.wiley.com/doi/full/10.4218/etrij.13.2013.0076*
[3] *https://blog.parsecgaming.com/description- parsec-technology-b2738dcc3842*
[4] *https://blog.parsecgaming.com/game-streaming-tech-in-the-browser-with-parsec- 5b70d0f359bc*
[5] *https://dev.to/giongto35/cloud-gaming-the-art-of-extremely-latency-streaming-477*
[6] *https://docs.microsoft.com/en-us/sysinternals/downloads/autologon*
[7] *https://medium.com/tensoriot/cloud-gaming-on-amazon-web-services-4be806c0051b*
[8] *https://docs.microsoft.com/en-us/azure/marketplace/cloud-partner-portal/virtual-machine/cpp-connect-vm#connect-to-a-linux-based-vm*
[9] *https://cloud.google.com/docs/games*
[10] *https://www.poppelgaard.com/amd-gpus-nvv4-series-vms-are-now-generally-available-in-azure*
[11] *https://azure.microsoft.com/en-us/global-infrastructure/services/?products=virtual-machines*
[12] *https://www.poppelgaard.com/amd-gpus-nvv4-series-vms-are-now-generally-available-in-azure*
[13] *https://support.microsoft.com/en-us/help/324737/how-to-turn-on-automatic-logon-in-windows*

VIII. Links for the Code and presentation

Google drive:
https://drive.google.com/drive/folders/1HYnK4QpjOLeYtqQlCdZI5roJt5W7jIJf?usp=sharing