National College of Ireland

Enhancing Cloud Gaming User Experience through Docker Containers in Fog Nodes

MSc Research Project
Msc. in Cloud Computing

Manoj Kannan
Student ID: x18131581

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Manoj Kannan
Student ID:	x18131581
Programme:	Msc. in Cloud Computing
Year:	2019
Module:	MSc Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	12/12/2019
Project Title:	Enhancing Cloud Gaming User Experience through Docker Containers in Fog Nodes
Word Count:	5996
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	12th December 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Cloud Gaming User Experience through Docker Containers in Fog Nodes

Manoj Kannan
x18131581

Abstract

The attraction towards cloud gaming has reached to a high intensity in modern days. The proliferation of gassed-up networks and the development in the field of cloud computing has attracted the researches to deep study and identify the gaps in Cloud Gaming to provide high-end quality of gamer experience. In time-horned cloud gaming model, the game scenes are rendered in a sophisticated game server and corresponding game video is generated. The generated game video will be encoded and sent over the network to the thin clients. The thin clients decode the encoded video and stream the video back to the players device. The fundamental requirement of gaming is to provide maximum quality of gamer experience. Cloud gaming suffers in terms of providing Quality of Experience (QoE), because the network transmission of game scenes from cloud game server to gamer device is distant. Since the traditional cloud gaming is deployed in virtual machine there is performance-overhead which also affects the QoE. We endeavour to minimize latency and increase performance in cloud gaming through this paper. The cloud game server will be offloaded to the fog nodes which is present at the edge network of the player based on Node selection algorithm. To increase the performance of cloud gaming, traditional virtual machine is replaced by light-weight containers. The proposed system achieves new methodology by deploying cloud game server through docker containers in fog nodes. Evaluation results has proved that fog-node based gaming has minimized the latency and increased the performance of cloud gaming.Fog assisted gaming has shown about 1.3MB increase in terms of TCP transfer rate over cloud gaming.

Keywords— Cloud gaming, Fog computing, Docker Container.

1 Introduction

Gaming industry has evolved all through the development of an Internet era. Early stage of the gaming era, the graphics rendering capabilities were low and also the Quality of user Experience (QoE) was not mediocre. Later as the field of gaming developed, we have advanced graphics with ultimate Quality of Experience for the user. In order to achieve high QoE in gaming, users are forced to upgrade their Graphics Processing Unit(GPU) which is quite expensive and gets outdated periodically.

In recent ages, cloud computing has engrossed the research community. Cloud Computing provides environment to perform high computations on powerful servers at low cost. It made researchers to have a deep dive and make the possible applications to move to the cloud environment. Both the providers and the users relish the benefits of cloud computing. Author Cai

et al. (2014) has discussed about providing the Gaming as a Service. He discusses about the benefits of adopting cloud gaming to both users and provides. This is because the Architecture of the Cloud game is designed in such a way that only the command emulator and video decoder will be present in the thin clients. On the other hand cloud game servers must have GPU rendering, video encoding and video streaming hardware support and provides support to the thin clients. Since all the powerful computations takes place in the cloud, players are relieved from having powerful hardware components.Author Chuah et al. (2014) discusses about the green solution of adopting the cloud gaming model. Author Cai et al. (2016) has debated the open challenges in Cloud gaming. Author states that, despite the great opportunity in cloud gaming, still there are several crucial challenges. One such Challenge is the latency issues which directly affects the QoE for the user. Since video streaming plays a vital part in QoE, the user and the cloud gaming server needs to be in sync recurrently. The delay in sync of this data affects the QoE.This delays is because of the distance between the cloud game server and the thin clients.Also the cloud gaming server uses virtual machines there is a significance performance overhead which also widely affects the user experience.

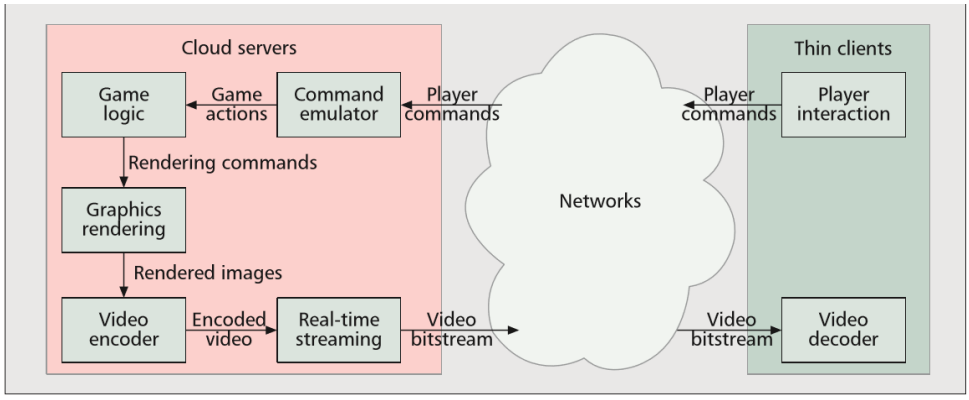

Figure 1: Workflow of the Traditional Cloud Gaming Architecture

Though Cloud Computing serves major advantage in the field of computing, it also suffers from its own drawbacks such as latency and bandwidth issues. This is mainly because of the distance between the cloud server and the user is located at a far distant. In order to reduce this latency issues researchers formed a new framework,named Fog nodes. Both Cloud and fog share most of the elements similar, but fog node will be located at the edge network closest to the user.This makes the application to be latency sensitive even to perform high computation tasks. All the data insensitive tasks can be offloaded from the cloud server to fog server.This additional node will help the cloud to reduce the latency issues.Also, the availability of the data is also high, since the data transfer over the network is less compared to the cloud. Another important break-through in the field of computing is virtualization using which virtual machine is created, but in order to perform high computational tasks virtual machines suffers from the performance-overhead issues.In recent years researchers paved a way for Docker containers, an replacement for the traditional virtual machine (Joy; 2015). Containers does not virtualize the entire Operating system,where as it virtualizes only the running applications. This makes the containers to improve the performance over virtual machines.

The proposed research aims to offload the cloud game server from cloud to fog which will enhance the user experience by reducing the latency issues. The traditional method of deploying the game server through virtual machines are replaced with Linux based Docker Containers. The Cloud game server decides which fog node to be selected based on the node selection algorithm. Node selection Algorithm calculates the servers with minimal latency based on the

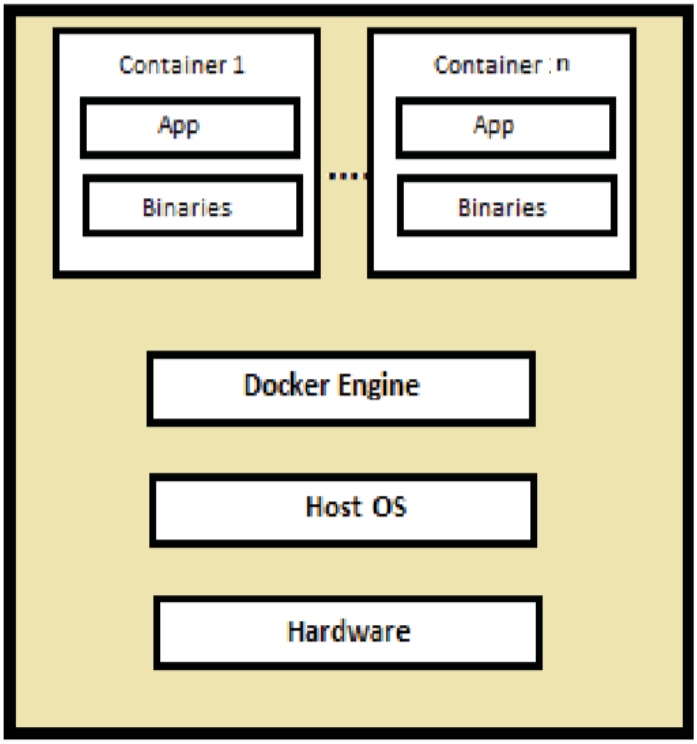

Figure 2: Docker Architecture

user location, GPU and hardware support to host game server.Thus, we propose Linux-based docker container in fog nodes which can achieve high Quality of Experience in cloud gaming than traditional Cloud Gaming.

Research Question:

Can performance of cloud gaming be improved by deploying docker container in fog nodes ?

Can the cloud game server be offloaded to fog nodes based on node selection algorithm to reduce latency in cloud gaming ?

The research aims to minimize latency and increase performance of the overall cloud gaming system. This is achieved by replacing traditional virtual machine into light weight containers.Node selection Algorithm identifies the fog nodes with minimal latency, located at the users edge network.Light weight frameworks are then deployed in computing fog nodes.Communication between the user and the fog nodes are established.

2 Related Work

Solution for the latency issues faced in cloud gaming: -

In Cloud Gaming the way of transmission of the encoded video and then decoding the video, affects the quality of experience due to the latency over the network. In order to minimize the response delay and the transmission delay author Lin and Shen (2016) has proposed a new architecture by recommending super nodes placing between the thin clients and the cloud game server. The paper proposes to offload the video encoding and decoding to the supernode where

as the computation takes place at the cloud game server. Author has approached the supernode selection based on the reputation score, the reputation score is calculated by taking the node with less transmission delay and the available capacity of the supernode. One drawback is that, due to the additional layer, many junk files can be transmitted over the network to the super node which can also affect the Quality of Experience for the player.

Author Choy et al. (2012) also discussed the way of solving the latency issues in the on-demand gaming. The author proposed an effective approach in solving the latency issues by offloading the computation to the nearest edge nodes which is present at the edge network of the players. Earlier in CDN the computations was not able to be performed. Author has enhanced the GPU rendering options in the CDN edge nodes and therefore the computations can be performed in the edge nodes. The evaluation of the proposed approach against the traditional approach was measured by performing players play against the EC2 instance versus the planet lab nodes which is the edge node and the end results proves that the Edge node minimizes the latency than the traditional approach.

Zhang et al. (2019) has approached the latency issues by placing the edge node but the node selection is achieved through the deep reinforcement learning. The author has adapted the sandbox technique in order to form the V-nodes. The virtual machines are replaced with the help of these Vnodes which reduces the play start-up delay experienced by the players. The workflow of proposed by the author is explained such that once the initial play request reaches the cloud game server, the game server identifies the nearest edge with minimal latency experienced by the player. Once the edge node is found the game request will be started from the edge node and the V-nodes are started based on the CPU/GPU rendering. The paper also proposed a Artificial neural network approach for streaming the video which enhances the Quality of Experience for the player. The neural network approach will adjust the frame bit rates based on the User Experience.

Author Cai et al. (2018) has approached the solution for the latency issues in cloud gaming through component-based ubiquitous Gaming system. Author has approached this by analysing the game players status and also by cognitively allocating the resources for different scenarios and different components of a game. Author has evaluated and the as per the outcome it is noted that well balanced way of allocating the resources to the cloud game server and the thin clients reduces the performance of the overall cloud gaming model.

Author Xu et al. (2014) proposed a effective algorithm which reduces the bit frame rate naturally. Homograph technique is used to predict the motion of the images in the video codec. Additionally the author also proposes a edge interpolation algorithm which is used to minimize the residues edge regions which can be used to perform the cloud gaming. Author uses x264 to test the proposed algorithm and the evaluation results displays that the proposed methodology 18% BD-rate reduction when compared with the traditional x264 approach.

Virtualization Advancement: -

The advert development in the field of virtualization made effective when the invention of sharing the GPU from the host to guest Operating system was invented. Author Qi et al. (2014) has proposed an effective way resource scheduling algorithm for cloud gaming. Author has approached this proposal by developing an API which enhances not only resource scheduling but also improves the performance of the On-demand gaming. Author has approached the solution with three scheduling algorithms, which also keeps the SLA metrics under the control. Based on the SLA metrics the scheduling algorithms works as per the requirement. Author has evaluated by using the real game scenes and has proved that VGRIS is capable of adapting to the virtual machine. Author has also discussed about the drawback in the system is that the author has majorly contributed towards the SLA metrics rather than the cloud gaming metrics.

Author Yadav and Annappa (2017) discusses the problem faced by the on-demand gaming where the current cloud gaming model has GPU allocation based on First come First server basis. This makes the traditional approach to fail from the SLA, there by they allocate ded-

icated GPU servers with sophisticated hardware support. In this paper author has proposed an Adaptive way of sharing the GPU which can run the dedicated gaming servers. Author has approached this algorithm with para virtualization technology and hook technique. After implementing author has evaluated the proposed performance by running Return to proxycon and fierly forest in 5 Virtual machine simultaneously. The proposed system shows 48 percent of the average frame rate where as the GPU is also shared by the 5 virtual machine. Author has set the resolution for only about 720p where as cloud gaming model runs with atmost 4k ultra resolution.

Light-weight Framework for Performance Optimization : -

The user experience in the cloud gaming is directly dependent on the latency and the performance of the system. Author Kämäräinen et al. (2015) has improved the overall performance of the cloud gaming by replacing the traditional virtual machine with the Operating system level name Containers. Author has evaluated the characteristics of a virtual machine and a container by having Gaming Anywhere as a base, which is a open source cloud gaming framework. X-server is shared among all the container which is used for security purpose. The performance of the CPU, GPU Memory usage and the game start time is measured among the QEMU VM vs the Linux based container. In which the Linux based containers outperformed and has showed a greater improvement when compared with traditional cloud gaming system.

Author Mondesire et al. (2019) analyses the new method through which the he has made the load balancing effectively. The proposed framework is the way of combining the containers inside the virtual machine. Based on the demand the way of allocating the resources such as GPU and memory of the overall system. Author has weighed the overall experiment results with four systems in comparisons namely three system are traditional method and the last one is the proposed system (. i.e.) combination of the virtual machine and containers. Out of which the proposed system lacks only in terms of Network bytes sent where as the proposed system outperformed in process memory. But the overall cost is being optimized in the proposed system. The anticipated system is also more secured when compared with that of the existing system.

The performance of the light weight framework containers against traditional virtual machine is analysed by the AuthorAhmed and Pierre (2018). Author has analysed the scalability and the deployment time in containers and virtual machine. Author has used the PostgreSQL database server as a back-end which is used for information retrieval to the front end and front end of the application is with Joomla PHP server. After evaluation author examined the results of the experiment and it is noted that containers outperformed well of handing 10000 request at a time where as virtual machine was able to handle only about 1800 request. For scalability comparison, word press application is used in the container-based application are deployed faster of about 22x times when compared to that of the virtual machine. Author has not compared about the security issues in both the systems.

Docker Container in Fog Computing Infrastructure: -

Deploying the docker container in fog computing infrastructure is widely studied by Author Raghavendra and Chawla (2018). Author has address the pain point in deploying the docker container in fog infrastructure such as the docker pull tasks takes longer duration. Author identified that the resources are under utilized during the docker pull because the pull request execute in parallel fashion. Author has made the pull request to be in a sequential manner . The proposed system had shown the improvement in the deploying the docker container in fog infrastructure by 4 times faster than traditional approach.

AuthorYin et al. (2018) has reviewed the need of the docker container in the fog infrastructure. The data that are transferred form the IOT devices has to be made highly available and has to be computed in a rapid fashion, thus docker container is used to process effectively, Additionally the portable feature in the docker container makes the computation to transfer the data faster than traditional approach. Author proposes a hierarchical approach by connecting

Approach	Performance	Latency	Gaming Over Cloud
Lin and Shen (2016)	**NO**	**YES**	**YES**
Choy et al. (2012)	**NO**	**YES**	**YES**
Zhang et al. (2019)	**NO**	**YES**	**YES**
Cai et al. (2018)	**NO**	**YES**	**YES**
Xu et al. (2014)	**NO**	**YES**	**YES**
Qi et al. (2014)	**YES**	**NO**	**YES**
Yadav and Annappa (2017)	**NO**	**NO**	**YES**
Kämäräinen et al. (2015)	**YES**	**NO**	**YES**
Mondesire et al. (2019)	**YES**	**NO**	**YES**
Ahmed and Pierre (2018)	**YES**	**NO**	**NO**
Raghavendra and Chawla (2018)	**YES**	**NO**	**NO**
Yin et al. (2018)	**YES**	**NO**	**NO**
Proposed System	**YES**	**YES**	**YES**

Table 1: State of the Art.

the IOT devices with the fog nodes and the fog nodes are also inter connected to each other. All these connections are controlled with the help of fog controller, which acts as a master node. By introducing the containers in the fog nodes high scalability is achieved for intensive tasks also the performance overhead is reduced.

Author Hoque et al. (2017) has approached the paper through a specific application and the use of deploying the application in the fog computing platform. The way of deploying the docker containers in the fog nodes makes the system to minimize the delay between the tasks and also increases the task scheduling. The overall process is divided into two. At first the device has to approve the task and second is to decide on which platform does the task has to be made (i.e .) on fog or cloud environment. A node is placed which decides based on the request evaluation algorithm which fog node can perform the computation by also keeping the resource constraint and the task scheduler decides where the task has to be assigned to fog or cloud environment. The experimental evaluation has shown that there is a 5 percent increase in the effective task scheduling also there is a difference in the overall execution time, which is because of the docker containers.

Author Hong et al. (2014) has discussed about the container orchestration in the fog computing infrastructure. Author discusses about the container orchestration tools, and the way of achieving the performance and scalability through the orchestration tools. He also discusses about the need of the container in the fog nodes. Author has evaluated the performance of the of docker container and it noted that about 0.036s container is faster than the traditional one. Author also evaluated the orchestration tools such as docker swarm, Kubernetes and Mesos-Marathon out of which docker swarm shows better performance while deploying containers in the fog infrastructure.

3 Methodology

In traditional cloud gaming architecture, the user inputs are captured and the captured input is transformed into commands through the command emulator. These commands are then converted into game events based on the game logic. The game events are rendered as the game video and the rendered video is encoded through a video encoder. The video is being encoded since it has to be sent back to the thin clients over the network. The encoded video will then be decoded in the thin clients. This way the traditional architecture suffers from the latency and

performance issues because of the far distance between the cloud game server and the player. This indirectly affects the Quality of Experience for the user.

The proposed methodology is inspired by the Author Lin and Shen (2016) by offloading the computations to the supernode which is located close to the user location. Author has made the decision to offload based on the reputation score of the player and the social server assignment. Author Kämäräinen et al. (2015) has made a proposal by replacing virtual machine into light weight containers. Though the Linux based container the game configurations was prepared. The combination of these two proposals is experimented in this research where containers in Fog nodes helps to maintain high QoE.

The proposal by the author Lin and Shen (2016) has not minimized the latency efficiently because the super nodes proposed is not located at the edge network of the user. Thus the latency can still be optimized from the proposal made by Lin and Shen (2016). Also the combination of deploying docker containers in Fog nodes for cloud gaming is not discussed in the state of the art. The proposed model engages Node selection algorithm in order to select the suitable fog nodes and deploy the docker containers in the selected fog nodes. The proposed architecture is implemented using Gaming Anywhere an open source cloud gaming platform, through which the game can be deployed. The Gaming anywhere system comprises of the command emulator, video encoder and video streaming codecs. The selection of the additional layer (. i.e.) Fog layer is determined based on the few checks which is written as a bash script. The Node selection is based on the steps explained below,

- Initial screening of which fog nodes needs to be selected is based on the minimal distance of which the Cloud gaming can have minimal latency compared to the traditional one (.i. e) the fog nodes are sorted based on the distance from the thin clients and the cloud as stated by the author Hong et al. (2014).
- Then the Quality of Experience threshold will be set and the fog nodes which meet the QoE threshold will be selected Dhib et al. (2016) .

In order to build the game server as a container, docker file has to be written. The docker file comprises of the Gaming Anywhere by author Huang et al. (2013) source code as a base and all the necessary packages are written in the docker file. From the docker file the container image will be created. In order to support the graphical display in container X-window is used using VNC desktop mode.

Once the fog node is selected based on the above checks, docker image will be deployed to the selected fog node. The docker container is allocated with the underlying resource of the fog nodes. The deployed container is the dedicated game server with the game resources such as video and audio encoder and decoder, Real time streaming service is pre-deployed.

By successful deployment of cloud game in docker containers and offloading the container to the fog nodes makes the proposed system to reduce the latency and enhances the performance of the cloud gaming. Game server is deployed as the docker image which also makes the deployment faster in the Fog node. The well-balanced Fog node selection and the proper deployment of the docker container in the fog nodes will make the Fog-assisted Cloud gaming system to reduce the latency also enhances the performance in a rapid fashion. Overall the Quality of Experience in cloud gaming for the player will be enhanced in the proposed architecture.

4 Design Specification

The Design specification of the proposed architecture includes multiple layers. First and foremost, layer for the cloud gaming is to set up a game server which has to be hosted in the cloud. AWS EC2 instance is chosen to host the game server in the cloud. The instance type chosen in EC2 was g3s.xlarge EC2 instance, because of the GPU support which was the most suitable instance to host the game server. Even though the EC2 instance has the GPU instance it was not having the graphics display to stream to the player. In order to obtain the video display in the EC2 instance a VNC connection was set up to a display monitor through which the videos can be streamed back to the thin clients. An open source cloud gaming framework Gaming Anywhere which was developed Huang et al. (2013)in the year 2013 was used as a game server. Since GA was developed in 2013 it has compatibility issues and the version issues. Initially the GA system was developed in ubuntu 12.04, but it has multiple compatibility issues. So to configure and set up the Gaming anywhere Ubuntu 16.04 operating system is taken into consideration. The Library packages for video streaming, encoding and decoding of the video also needs to be installed. The libraries required such as FFMPEG, libx264, libx265, libvpx and so on. All these libraries are pre-compiled in the docker container image. A docker files is written with all the installation of libraries, dependencies and the packages. Along side the installation of the Gaming Anywhere server is also wriiten in the docker file. A docker image is created with the help of the docker file . The developed docker image is then pushed to the docker hub. The components required to set up a game server with Gaming Anywhere is

- libx264 : It is a open source software which is used to encode the video streaming service.
- X-Server: To run the graphical applications inside the container.
- Nvidia Driver: To support the GPU acceleration.
- Docker CE Debian package: - To support docker containers in ubuntu.
- libva1 - An API for video accelerations.
- libasound 2 :- For Audio enhancement and sound management in the ALSA library
- libswscale 2 :- For video scaling and rendering.
- libxtst6 :- For client interface x-window.

5 Implementation

The implementation of the proposed system comprises different phases and each of the phase is explained below.

- Docker image creation
- Cloud game server
- Fog server identification and resource allocation to container
- Offloading to the Fog server
- Communication between the Fog server and the thin clients

Docker image creation:
Initial stage for the implementation is to create a docker image with all the required libraries and packages installed. Since the docker container does not support graphical display the X-window needs to be installed in the docker file for creating the docker image with graphical support. Additionally, the docker image should also have the NVidia support graphics which supports the GPU graphics acceleration along with the image. All the game configurations has to be pre-build inside the docker file.

Cloud Game Server:
A cloud server with Ubuntu machine as a base along with the GPU support needs to be launched. The launched cloud server needs to have the container package installed along with the other networking packages to be installed. The connection for the traffic outbound and inbound rules needs to be modified according to the client traffic flow.

Fog Node Identification:
Once the play request is reached the game server, game server will fetch the IP address of the player and locate for the edge node based on the algorithm proposed in the paper Hong et al. (2014). The proposed algorithm not only takes the Fog server identification as the only task but also to identify the Fog server without allowing the players to degrade the Quality of Experience. The formulation to calculate Quality of Experience Dhib et al. (2016)is given below

$$Q(x,p,v) = \gamma 1 Nx + \gamma 2 D(p,v) \text{ —— } 1$$

Where Network delay function is denoted by Nx and D(p,v) is the processing delay and $\gamma 1$ and $\gamma 2$ are parameters. If the Quality of Experience is greater than the threshold of the QoE then add the fog nodes to the selected list. Once the selection list is prepared, then for each of the selected fog servers calculate the server location which is present nearest to the player location with minimal latency Hong et al. (2014). By achieving this the 95 percent of the quality of experience for the user is achieved.

Algorithm 1 Node Selection Algorithm

Require :
p: number of players, QoEThr: trheshold of minimal QoE requested
1: **for** each player x ϵ 1..p **do**
2: calculate the QOE function Q(x,p,v)
3: **if** Q(x,p,v) $\geq$ QoEThr then add the server to possible selection list
4: **endfor**
5: **for** each possible selection
6: **sort** servers on network latency to p in asc. order
7: select the server node with minimal latency
8: **endfor**

Offloading to the Fog Server:
Once the server has been identified the docker image which contains the game server and the game configurations are offloaded and deployed at the Fog server and the game server kick starts from the selected Fog node. This makes the proposed gaming model to minimize the latency from traditional model also since the docker image is created and offloaded all the pre-requisite for the game will be installed which makes the deployment of the server 10x times faster compared to the traditional deployment.

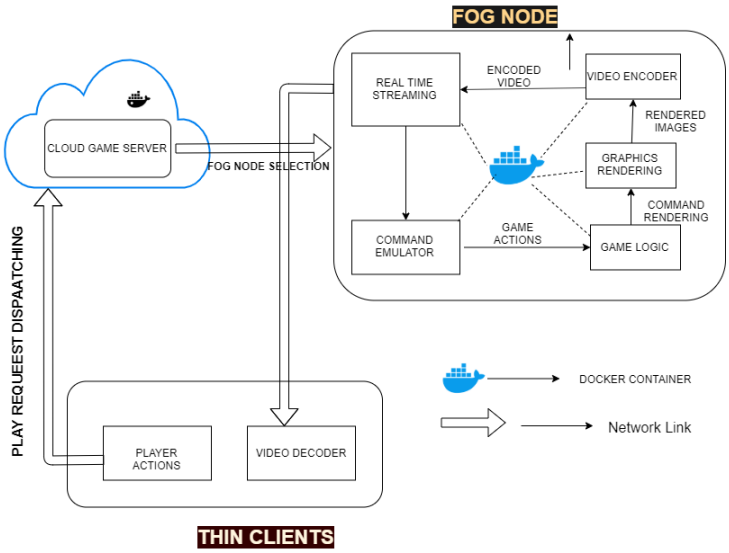

Figure 3: Workflow of the proposed Gaming Architecture

Communication between the Edge Server and the Thin Clients :
Once the game server starts the IP address of the game server will sent over to the client and the user actions from the client device are transferred as commands and are sent to the newly got Fog server IP address, This establishes the communication to stay in between the Fog server and the player. Additionally, all the user data are uploaded simultaneously to the cloud server so that the continuity of the user profile will be maintained game server.

The work flow of the proposed architecture is explained in the below steps.

- Initial play request will be dispatched from the thin clients to the cloud game server.
- Cloud game server identifies the Public IP address of the player and sends as an input to the node selection algorithm.
- Node selection algorithm selects the fog nodes with minimal latency to the user by also meeting the Quality of user experience threshold limit.
- Fog Node is selected at the edge network of the player and the command to deploy the container image will be sent to the fog nodes.
- After the container image is deployed, the game server will be ready to host the game in fog nodes.
- The fog nodes receives the public IP of the player from cloud game server, therefore the play request will be dispatched to the thin clients and the communication will be established.
- Once the communication is established, the gamer starts to play the game , and the player actions are sent over the network to the fog nodes. Command emulator in the fog nodes transforms the user actions into commands.
- The transformed commands are executed as per game logic. Once the game logic is applied, the game videos are generated through graphics rendering process.
- Game video will then be encoded through the video encoder and are transferred to the video streaming service.

- The video streaming service sends the encoded game video to the thin clients and the encoded video will be decoded and the game video will be streamed back to the player device.

6 Evaluation

6.1 Experiment / Case Study 1

The proposed research is implemented and tested by considering AWS EC2 GPU instance as the cloud game server. The first-person shooter game Assault cube is considered for the test case. Gaming Anywhere an open source gaming framework which provides the source code of the game. The game server is deployed in the AWS Ohio region and the Fog node is considered as AWS EC2 GPU instance Ireland region. All the dependencies of the game is developed as a docker file. Additionally the other configurations to build a docker file with graphical windows and the network configuration to accept the in bound and out bound docker and pushed to the docker hub. The game request is dispatched from the thin clients located in the Ireland to the game server located in the Ohio region. The game server identifies the location of the player and selects the fog node located at the edge network to the player. Game server then sends the command to initiate the play request along with the docker image to the selected fog node. The docker image is deployed in the fog nodes and the game is initiated. To evaluate the performance and the latency of the traditional cloud gaming and the proposed approach IPERF and PING commands are used. Additionally for CPU performance Phoronix test suite bench marking tool was used.

Initially the game is made to run on Cloud servers and the result is evaluated and the same game is offloaded to the edge server and the game kick starts from the edge. Both the results are examined and the results are displayed below figure. The results are taken by using the IPERF command from the client machine to the cloud and Fog servers. TCP transfer rate is identified in Figure 4 through which the transfer bytes for fog server was high compared to cloud. On the average fog node transferred 14.9mb per transfer where as cloud transfer rate was only upto 13.2mb. This proves that the proposed edge system has outperformed in terms of TCP transfer rate thus reducing the latency in the overall cloud gaming model. .

Further to study the performance of the cloud gaming system, the CPU performance was measured. Phoronix test suite benchmark tool was used to study the CPU performance by running the Assault cube game in traditional EC2 instance versus Linux-based docker containers. On examine the chart in figure 5, docker container has shown improvement in terms of CPU performance when compared to virtual machines. Additionally docker CPU performance does not vary significantly where as in virtual machine the performance varied.

6.2 Discussion

The proposed system is implemented using Gaming Anywhere open source framework which was developed in 2014. Since then, the advancement in video encoding and decoding has grown where as Gaming Anywhere still supports only older version. This makes the gaming framework outdated. So much efforts were spent in terms of rebuilding the Gaming anywhere framework. As the advancement in the virtualization is at the peak, efforts can be spent in terms of virtualizing GPU resources for containers through Kubernetes device plugin can be studied. This makes cloud gaming more efficient in terms of allocating resources to containers.,

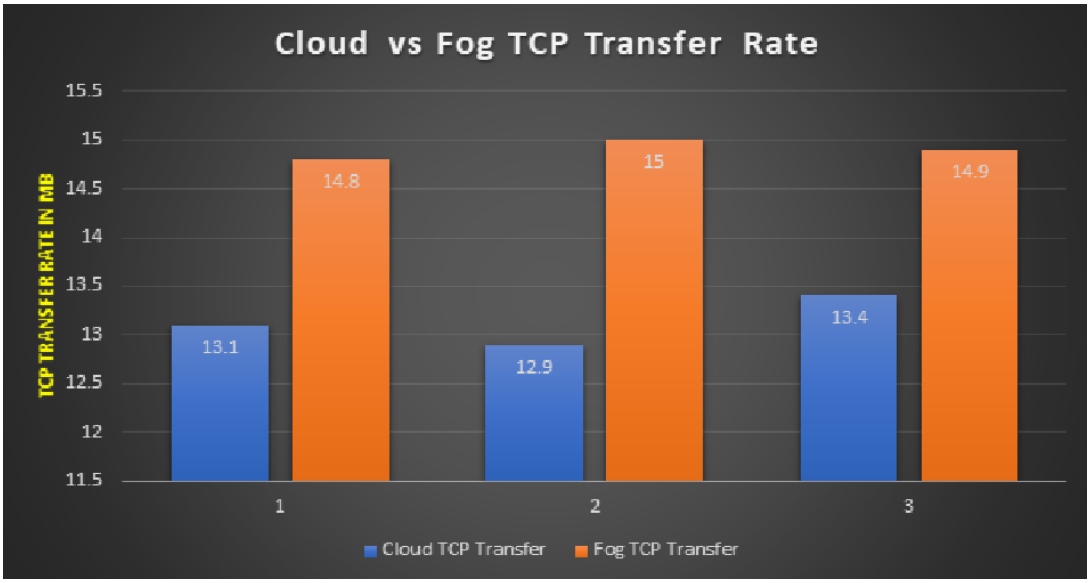

Figure 4: Cloud vs Fog TCP Transfer

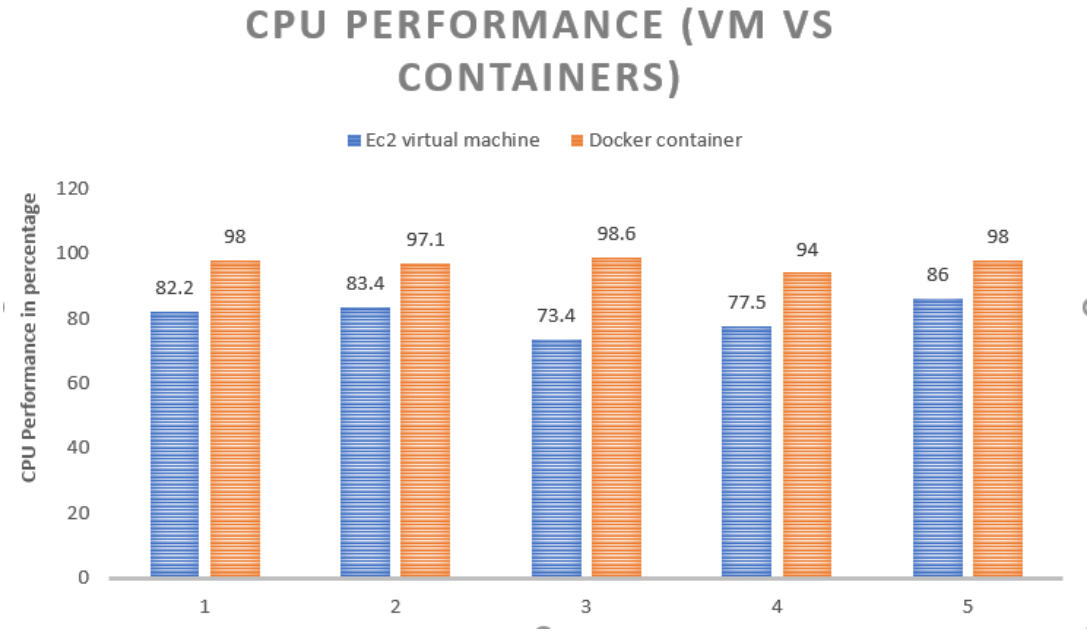

Figure 5: CPU Performance evaluation Virtual Machine vs Containers

7 Conclusion and Future Work

The proposed system is to minimize the latency in the cloud gaming by offloading the cloud game server into suitable fog nodes with the help of docker containers. The overall result after implementation has proved that the Fog assisted cloud game increases the quality of experience for the user.The improvement of the CPU performance is studied after deploying the game server in docker containers.The end result of the proposed system is to improve the Quality of User Experience which is achieved by the help of fog nodes and docker containers. Future work can be made in terms of allocating virtualized resources to the GPU. One or more containers can share a single GPU by the help of kubernetes using Device plugin. Further the study can also be in improving the Node selection algorithm by dynamically allocating containers based

on the available regions.

References

Ahmed, A. and Pierre, G. (2018). Docker container deployment in fog computing infrastructures, *2018 IEEE International Conference on Edge Computing (EDGE)*, IEEE, pp. 1–8.

Cai, W., Chen, M. and Leung, V. C. (2014). Toward gaming as a service, *IEEE Internet Computing* **18**(3): 12–18.

Cai, W., Chi, Y., Zhou, C., Zhu, C. and Leung, V. C. (2018). Ubcgaming: Ubiquitous cloud gaming system, *IEEE Systems Journal* **12**(3): 2483–2494.

Cai, W., Shea, R., Huang, C.-Y., Chen, K.-T., Liu, J., Leung, V. C. and Hsu, C.-H. (2016). A survey on cloud gaming: Future of computer games, *IEEE Access* **4**: 7605–7620.

Choy, S., Wong, B., Simon, G. and Rosenberg, C. (2012). The brewing storm in cloud gaming: A measurement study on cloud to end-user latency, *Proceedings of the 11th annual workshop on network and systems support for games*, IEEE Press, p. 2.

Chuah, S.-P., Yuen, C. and Cheung, N.-M. (2014). Cloud gaming: a green solution to massive multiplayer online games, *IEEE Wireless Communications* **21**(4): 78–87.

Dhib, E., Boussetta, K., Zangar, N. and Tabbane, N. (2016). Modeling cloud gaming experience for massively multiplayer online games, *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, pp. 381–386.

Hong, H.-J., Chen, D.-Y., Huang, C.-Y., Chen, K.-T. and Hsu, C.-H. (2014). Placing virtual machines to optimize cloud gaming experience, *IEEE Transactions on Cloud Computing* **3**(1): 42–53.

Hoque, S., de Brito, M. S., Willner, A., Keil, O. and Magedanz, T. (2017). Towards container orchestration in fog computing infrastructures, *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2, IEEE, pp. 294–299.

Huang, C.-Y., Chen, D.-Y., Hsu, C.-H. and Chen, K.-T. (2013). Gaminganywhere: an open-source cloud gaming testbed, *Proceedings of the 21st ACM international conference on Multimedia*, ACM, pp. 827–830.

Joy, A. M. (2015). Performance comparison between linux containers and virtual machines, *2015 International Conference on Advances in Computer Engineering and Applications*, IEEE, pp. 342–346.

Kämäräinen, T., Shan, Y., Siekkinen, M. and Ylä-Jääski, A. (2015). Virtual machines vs. containers in cloud gaming systems, *2015 International Workshop on Network and Systems Support for Games (NetGames)*, IEEE, pp. 1–6.

Lin, Y. and Shen, H. (2016). Cloudfog: leveraging fog to extend cloud gaming for thin-client mmog with high quality of service, *IEEE Transactions on Parallel and Distributed Systems* **28**(2): 431–445.

Mondesire, S. C., Angelopoulou, A., Sirigampola, S. and Goldiez, B. (2019). Combining virtualization and containerization to support interactive games and simulations on the cloud, *Simulation Modelling Practice and Theory* **93**: 233–244.

Qi, Z., Yao, J., Zhang, C., Yu, M., Yang, Z. and Guan, H. (2014). Vgris: Virtualized gpu resource isolation and scheduling in cloud gaming, *ACM Transactions on Architecture and Code Optimization (TACO)* **11**(2): 17.

Raghavendra, M. S. and Chawla, P. (2018). A review on container-based lightweight virtualization for fog computing, *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, IEEE, pp. 378–384.

Xu, L., Guo, X., Lu, Y., Li, S., Au, O. C. and Fang, L. (2014). A low latency cloud gaming system using edge preserved image homography, *2014 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, pp. 1–6.

Yadav, H. and Annappa, B. (2017). Adaptive gpu resource scheduling on virtualized servers in cloud gaming, *2017 Conference on Information and Communication Technology (CICT)*, IEEE, pp. 1–6.

Yin, L., Luo, J. and Luo, H. (2018). Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing, *IEEE Transactions on Industrial Informatics* **14**(10): 4712–4721.

Zhang, X., Chen, H., Zhao, Y., Ma, Z., Xu, Y., Huang, H., Yin, H. and Wu, D. O. (2019). Improving cloud gaming experience through mobile edge computing, *IEEE Wireless Communications* .

8 Appendix

8.1 AWS Instance setup

For experimental setup we have chosen Amazon EC2 instance with Ubuntu as a underlying operating system.

. **Step 1** :- Select AWS EC2 instance with Ubuntu 16.04 AMI

Step 2 :- Select the GPU instance to support graphics acceleration.

Step 3 :- Configure the Inoubound and outbound security connections to support the traffic flow.

Step 4:- Launch the AWS instance

8.2 Install Docker Container

Step 1 :- Update the packages in the ubuntu EC2 instance created.

sudo apt-get update

Step 2 :- Remove the older version of the docker if present.

sudo apt remove docker-engine docker.io

Step 3 :- Install the docker container

sudo apt install docker.io

Step 4 : - Start and enable the docker engine

sudo systemctl start docker

sudo systemctl enable docker

sudo docker version.

8.3 Configure Docker Container for Cloud Gaming System :-

Step 1:- Set up docker graphical desktop to support cloud gaming. To support the graphical application we need to allow the X-server in the container. Also install the required packages.

sudo apt install xeyes

Also while buildingruning the docker container the special arguments has to be set.

sudo docker run –net-host –env=”DISPLAY”

Step 2 :- Install the Nvdia graphics driver to support the GPU accleration in the cloud gaming.

sudo apt install NVIDIA driver

Run the graphics driver by sh /tmp/NVIDIA-DRIVER.run
This will install the NVIDIA driver inside the docker contaier which will enhance the GPU accleration in the docker containers.

8.4 Configure the Gaming Anywhere Frameworkinside docker container

Step 1 :- Create a docker file which has to contain all the necessary installation setups to build a docker image of Gaming Anywhere open source frame work.

vi dockerfilename

Step 2 :- Since Gaming Anywhere setup is having dependencies issues, the compiled version of Gaming Anywhere server framework is placed in the github repository given below.

https://github.com/CloudMaster-Manoj/gaminganywhere.git

Above snippet in Figure 4 is the steps involved in creation of the docker file.

Step 3 ;- Once the docker file is created docker image has to be build from the below command

sudo docker build -t reponame/tag .

Step 4 Once the docker image has been build successfully,the docker image has to be pushed into the docker hub.

```
FROM ubuntu
RUN apt update
RUN mkdir myprog
WORKDIR /myprog
ENV DIRPATH /myprog
RUN pwd
RUN apt install -y libxext6 libxext-dev libxtst6 libxtst-dev libfreetype6 libfreetype6-dev libogg-dev yasm nasm libx11-dev mesa-utils libpul
 libpostproc-dev libavformat-dev libsdl2-dev libsdl2-ttf-dev libopus-dev cmake ffmpeg git libx265-dev libsdl-image1.2 vim wget
RUN apt-get install -y patch make cmake g++ pkg-config libx11-dev libxext-dev libxtst-dev libfreetype6-dev libgl1-mesa-dev libglu1-mesa-dev
RUN apt-get install -y libsdl2-dev libsdl2-ttf-dev yasm \
                libmp3lame-dev libopus-dev libogg-dev \
                libvorbis-dev libtheora-dev \
                libvpx-dev libx264-dev libxvidcore-dev

RUN ln -s /usr/include/locale.h /usr/include/xlocale.h
RUN git clone  "https://github.com/CloudMaster-Manoj/gaminganywhere.git"
RUN cd /myprog/gaminganywhere/ && \
    pwd && \
     /bin/bash -c "source /myprog/gaminganywhere/env-setup"
RUN cd /myprog/gaminganywhere/deps.src/ && pwd &&  make
RUN apt install -y libavdevice-dev libavfilter-dev
RUN chmod 777 /etc/ld.so.conf
RUN cd /myprog
RUN export LD_LIBRARY_PATH="/myprog/gaminganywhere/deps.posix/lib"
RUN echo "/myprog/gaminganywhere/deps.posix/lib" >> /etc/ld.so.conf
RUN apt install -y ffmpeg
RUN pwd && ls -lrt
#RUN mkdir test && pwd && cd test && ls -lrt && pwd && ls -lrt &&  git clone "https://github.com/CloudMaster-Manoj/gaminganywhere.git" && ls
#RUN rm -rf /myprog/gaminganywhere/ga/module/server-ffmpeg/
#RUN rm /myprog/gaminganywhere/ga/module/server-ffmpeg/server-ffmpeg.cpp
#RUN mv /myprog/test/gaminganywhere/ga/module/encoder-x264/encoder-x264.cpp /myprog/gaminganywhere/ga/module/encoder-x264/encoder-x264.cpp
#RUN mv /myprog/test/gaminganywhere/ga/module/server-ffmpeg/server-ffmpeg.cpp /myprog/gaminganywhere/ga/module/server-ffmpeg/server-ffmpeg.c
#RUN ls -lrt /myprog/gaminganywhere/ga/module/encoder-x264/
#RUN ls -lrt /myprog/gaminganywhere/ga/module/server-ffmpeg/
#RUN rm /myprog/gaminganywhere/ga/client/Makefile
#RUN cp /myprog/test/gaminganywhere/ga/client/Makefile /myprog/gaminganywhere/ga/client/
#RUN rm /myprog/gaminganywhere/bin/config/server.assaultcube.linux.conf
#RUN cp /myprog/test/gaminganywhere/bin/config/server.assaultcube.linux.conf /myprog/gaminganywhere/bin/config/
RUN ldconfig
RUN pwd && ls -lrt
RUN apt update
RUN apt install -y libga-dev
RUN cd /myprog/gaminganywhere/ga/module/server-ffmpeg && mkdir ffmpeg
RUN mv /myprog/gaminganywhere/ga/module/server-ffmpeg/ffmpeg-1.1 /myprog/gaminganywhere/ga/module/server-ffmpeg/ffmpeg
RUN cp -r /myprog/gaminganywhere/ga/module/server-ffmpeg/ffmpeg/ffmpeg-1.1/*  /myprog/gaminganywhere/ga/module/server-ffmpeg/ffmpeg/
RUN ls -lrt /myprog/gaminganywhere/ga/module/server-ffmpeg/
#RUN /bin/bash -c "source /myprog/gaminganywhere/env-setup"
#RUN rm /myprog/gaminganywhere/ga/server/Makefile
#RUN cp /myprog/test/gaminganywhere/ga/server/Makefile /myprog/gaminganywhere/ga/server/
```

Figure 6: Docker file creation

```
#!/bin/bash

echo -n "Enter the Total number of IP address:"
read n
echo "Enter the IP address :"

i=0
while [ $i -lt $n ]
do
    # To input from user
    read a[$i]

    # Increment the i = i + 1
    i=`expr $i + 1`
done
echo "Output :"
i=0

echo -m "Enter the client IP address"
read m


while [ $i -lt $n ]
do

    #ip= curl https://ipinfo.io/${a[$i]} | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" country
        ip= curl https://ipinfo.io/${a[$i]} | grep country
        hostip= curl https://ipinfo.io/$m | grep country
        echo $ip
        echo $hostip
        if [ $ip == $hostip ]
        then
                echo "success"
                echo ${a[$i]}
        fi
        i=`expr $i + 1`
done
~
~
~
~
```

Figure 7: Node Selection Algorithm

8.5 Offloading the cloud game server to Fog nodes through Docker Container

Step 1 :- Involves the identification of Fog nodes located at the edge server. This is selected with the help of Node selection algorithm.

Once the Fog server is selected, docker image has to be pulled from the docker hub.

docker pull imagename

Once the docker image is pulled the game server engiene starts the deployment of the game.

8.6 Client Configuration

: -

Step 1 :- Download the prebuild Gaming Anywhere file from the below link

http://gaminganywhere.org/download.html

Change the client configuration based on the Game setup

Step 2 :-
Client game has to be configured and the RTSP client has to be opened to accept the incoming video streaming from fog server.

Step 3:- Set-up X-server VNC desktop connection to support the streaming service.

8.7 Game play start

Final step is to run the game with the below command

./ga-client config/client.rel.conf Ip-address-of-selected-fognode/desktop