Cloud Gaming System in Docker Container Image

MSc Research Project
Cloud Computing

Arun Pugalendhi
Student ID: X17127874

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Arun Pugalendhi
Student ID:	X17127874
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Divyaa Manimaran Elango
Submission Due Date:	20/12/2018
Project Title:	Cloud Gaming System in Docker Container Image
Word Count:	XXX
Page Count:	17

Signature:	
Date:	28th January 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):	

Cloud Gaming System in Docker Container Image

Arun Pugalendhi
X17127874

28th January 2019

Abstract

Cloud gaming is the future technology for the gaming industry and it is a cutting edge technology. This is developed based on the bases of gaming on demand. with this application the user can play the game anywhere and any time. the main feature of this application is, it can be used from any device, which is much more scalable. The main challenges in the cloud gaming handling the resource based on the user requirement or Resource allocation and QoE. these problems Arieses because each and every game has its own requirements, which will differ on the bases of game-play also, by which this problems lead to improper resource allocation, by which this will lead to loss in QoE and the over all performance. In this paper it explains how to improve the performance of the cloud gaming by initializing the cloud gaming package inside the containers which will allow the application to be more reliable by secularizing its resource allocation and increasing the overall performance of the cloud gaming system and makes it more reliable with utilizing less resource.

Contents

1 Introduction

Cloud gaming is the current technology which was developed based on the gaming on demand. Cloud gaming is divided into 2 types which is based on video streaming and file streaming. Hence most of the gaming instance will be moved to the cloud. It will be an advantage for the users where they can play the games from anywhere like PC, mobile and TV using a thin client. This technology allows the user to access the games without owning the gaming console or highly configured gaming PC. Hence it is totally depended on the internet, which is the most challenging part. This enables the user to access the games from multiple devices which doesnt have high hardware setup. It is well known as gaming as service or gaming on demand. Hence through this technology the users can access any type of game from anywhere, any time and from any devices. The user can access the technology thin client. The thin client can be accessed from a less powerful device, from the thin client user will be able to play games based on the subscription. In the thin client the game logic will be applied from that the signals will be decoded and will be transmitted to the cloud server.

In the cloud gaming system, the user will access the service through a thin client, which is provided by the service provider. The user will be able to play his requested game, based on his subscription and requirements the resource will be allocated from the server side. When the user plays the game, the gaming logics will be will be encoded and will be transmitted to the server through internet. In the server the logics will be decoded and it will be applied on the game. Thus, the game will run based on the inputs and the video/audio is captured. The captured video and audio is encoded through an encoder and it is transmitted through real time protocols, where it helps to transmit the video as an encoded format. In the client side the video will be decoded and it will be streamed in the user end. The user will able to see the output in a video streamed format. Hence the user doesnt need to upgrade their in-house resources and they can play any new gen games without buying it or creating any dedicated system for it. Thus, this technology will reduce the CAPEX and OPEX in both server and client, hence this paper talks about the development of cloud gaming system which is enabled through containers. A cloud gaming system will be developed inside the containers instance. This helps by reducing the value, responsibility, keeps the use updated, a lot of scalable and fewer resource utilization. Virtualization plays a serious role within the cloud gaming wherever within the server aspect the resources are vitalized supported the user necessities. Hence the most challenging task is resource scheduling and performance enhancement through handling the GPU and FPS of the system. There are many problems faced in this area. This differs based on each game, where each and every game has its own requirements, because the game developer, develops the game by thinking that the total resource present in the system is for the particular game and develops it based on it. By which the resource utilization varies by each and every game. The total quality of the game also depends upon it.

Containers are OS level virtualization which allows the user to run an application on top of it without any OS. Containers are more scalable and it is more reliable. Containers reduces the cost and it can increase the usage instance without any issue. With less resource large number of users can access the resource using containers. In this paper we are developing a cloud gaming system with the help of gaming anywhere application Huang et al. (2013)Where the system will be created inside the containers, thus this helps to increase the overall performance of the cloud gaming system. This a container is developed on top of Host OS and the cloud gaming system is executed inside the containers. This cloud gaming system is developed using multiple library functions. Hence this system will help to increase the overall performance of the cloud gaming system and helps to reduce the overall cost and by updating the users regularly with more scalability and less resource utilization

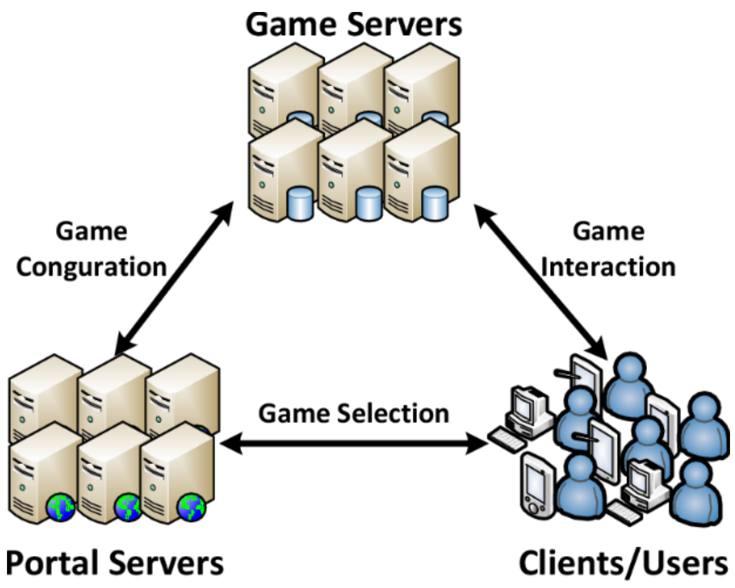

Figure 1: Cloud gaming system

1.1 Contribution

Our Contribution

The proposed solution is for increasing the overall performance based on frame rate and resource of the cloud gaming system. The solution provides

- There will be an overall increase in the FPS and the latency of the system
- The required resources will be scheduled properly based on the user requirements and the gaming instance
- The overall performance of the system will be enhanced by implementing the system inside the containers technology

1.2 Use Case Scenarios

the main focus is to reduce the performance degradation, lower FPS rate which is caused due to lower latency and improper resource allocation

- FPS (Frames per second): The quality of the game depends upon the FPS rate, when the system renders higher rate of FPS the quality of the game will be high. FPS determines the QoE (Quality of Experience) of the game. These QoE will be determined by the focus of the player in the game. hence due to latency problem the FPS drops in each and every frame. if there is problem in the instance or with the resource also the FPS of the game will drastically drop

- Resource Allocation: Each and every game has its own resource requirements and workloads or even a single game it will be having a different workload based on the game-play. Because a server may not be able to run a n number of high graphics games, it is fair to run a single high graphics game and n number of low graphics game in the server, but when many number of users go for high graphics game the server is not able to allocate perfectly and it is not able to be more sustainable as per the suggestion of (Chen et al.; 2014).

Hence in this paper we propose a cloud gaming system which will be implemented inside the containers to increase the overall performance.

2 Related Work

The author (Zhang et al.; 2016) proposes a specially designed cloud gaming system called G cloud which follows a hybrid clustering system for scheduling the CPU/GPU. It basically depends on user-level virtualization, which helps to enhance the cloud gaming experience. Hence the G Cloud system defines the cloud gaming into 2 types which is CPU critical and memory-IO-critical which totally relies on the total capacity consumed by the game from server. Here the proposed G cloud builds a user-level virtualization which helps to run the gaming instance separately which runs in a single server from other servers and consumes the outputs of the game and streams it to the user end. Here they are fixing the FPS and to allocate the right amount of resource to perform well.

Here the author (Wang et al.; 2016) explains the barriers in the CPU optimization with high Quality of Services. The author is proposing 3 algorithms which helps to over come the barriers. The algorithms are SLA-aware scheduling for handling the GPS resource precisely, Open loop control algorithm for handling the CPU and Adaptive control algorithm for CPU scheduling. These algorithms are developed based on the resource allocation without compensating with the QoS.

Where else (Yadav and Annappa; 2017) explains the problems the improper scheduling process in GPU virtualization in the datacentres. here the author proposes a adaptive scheduling mechanism to handle the GPU resources properly. Hence the adaptive scheduling system is developed based on scheduling controller, monitor, VM scheduler, VM list, Process ID and involves in interacting with the GPU host (Yadav). By implementing these operations helps to give detailed review on running VM and its resource allocation process.

(SanWariya et al.; 2016) tries to analyse the processing overhead of 3 type zero hypervisors, hence the 3 type zero hypervisors are KVM, Hyper-V and ESXI. Thus, the overall performance for these hypervisors is evaluated in the game on the bases of Graphic benchmarking and FPS rate. To gain a better result the author uses a average FPS rate and achieves that the ESXI is the best hypervisor for cloud gaming system for getting results based on reliability, QoS and Scalability.

The author (Shea et al.; 2015) proposes a new design to create a connection between online and cloud gaming system by implementing it in fully virtualized cloud gaming system. Thus, it analysis many design flow which exist in cloud gaming system. It suggests methods to improvise the performance of the hardware and software encoding. Thus, the author concludes that the hardware encoding improves the overall cloud gaming experience rather that software encoding.

FGCG is a new cloud gaming design which was proposed by (Zhang et al.; 2018) to enhance the high resource consumption on CPU-GPU cluster. This system works on the basis segregating the gaming workloads into small tasks, by which this task can be continuously segregated to multiple devices. Hence a trace driven simulation was implemented which helped to recognize that the FGCG system improvises the utilization of the resources in the system when compared to other cloud gaming systems.

The author (Usman et al.; 2017) proposes a architecture implement the unavailable bandwidth which is required to support the remote rendering system. Thus, the author uses the AWS cloud to implement the remote rendering process by using a game for testing. Thus, the outcome of the process, it was able to transfer on 1kbps of data between the player and cloud. Hence the author improvises the system by implementing the system with 3 layers client end, cloud end and data between the game client and the cloud service. Hence these layers are used to run the game which gives initial information from client end through cloud end and the information is again transmitted between client and the cloud. By which this model helps to load, maintain and scale the gaming instance from anywhere. Where else the remote rendering process is not applicable in this process.

To enhance the overall performance the author (Hong et al.; 2015) proposes a trace driven simulation and a heuristic algorithm, where the author calculated the outcomes using this simulation test. Based on the user requirements for the highest QoE there are multiple VMs are installed in the server. The algorithm follows QDH where it allocates the server for the users and sorts it in ascending order to perfectly allocate the right and the required VM to the required user which has the competence to run the system perfectly.

The author (Zhang et al.; 2014) proposes an end to end latency improvising technique in the data centre, which is developed using a OpenFlow controller. By which the algorithm called as Lagrangian Relaxation time efficient heuristic algorithm, hence which reduces the end to end variation and delay.

VGASA is a resource scheduling mechanism which was proposed by (Qi et al.; 2014). This VGASA concept follows 3 scheduling algorithms, the algorithms are SA, FSA and ESA. By which the SA algorithm achieves the SLA parameters of a running game. The FSA algorithm is used for allocating GPU resource to the required game and avoids any loss in resource, by which has a better performance than the SA. The ESA allows the instance to run the game in a highest FPS rate with more GPU utilization.

(Kmrinen et al.; 2015) scheduling algorithm called vGRIS. vGRIS is a virtualized GPU isolation and Scheduling in cloud computing management framework. the scheduling is done through various workloads by using the set of APIs initiated for designing the scheduling algorithm. Hence this is the light weight resource scheduler between the client and the server, by which the VMs like vmware and VirtualBox only shares single GPU. This scheduling process interrupts the libraries of the GPU instead of VMs. This system has 3 scheduling algorithms. The algorithms are SLA aware scheduling, hybrid scheduling and proportional scheduling. To avoid the SLA breaching, SLA aware scheduling is used which allocates the required amount of GPU to the VM. In the proportional share scheduling the total GPU resources are allocated to the active VMs. in the hybrid scheduling it has both characteristics of the previous algorithm, hence these concepts can be used for the maximum utilization of the resources.

Based on the various factors the SLA sharing between the client and the service provider this strategic cloud environment is developed which is based on the interaction between multiple clients and multiple server in the server environment which was proposed by (Zhang and he Zhou; 2017). Based on the users desire a Nash equilibrium is achieved and the resource allocation is being handled through Bayesian strategic game. thus, this system allocates the SLA based on the task and criteria.

The author (Wang et al.; 2009) describe the values of container than the virtual machines increase the performance overall, by which the original performance of the instance will be increased. Hence by this the author suggest that the cost can be minimised and the will be able to increase the maximum number of instances.

(Seneviratne and Leung; 2011) enhances the energy efficiency by two methodologies. One is based on non-cooperative game where it will be used for developing the resource allocation process which is located inside the cloud with the help of NG-TSRA resource allocation algorithm. To achieve the Nash equilibrium the process should be implemented in iteration algorithm.

More frequently than not, different end user systems have been a part of the cloud gaming system. This set of different end user systems have helped in improving the network quality that influences the experience of gamers in a positive manner. Keeping this in mind, this project proposes a component-based gaming system that is equipped with cognitive abilities that learn about players and optimizes itself and allocates resources for the software units in a game. This paper (Cai et al.; 2018) concludes that a well-balanced software segment will guarantee better performance with cloud gaming.

A Buyer status and content aware packet scheduling was proposed for the enhancement of video quality and playback continuity. The client buyer is predicted by making use of creation of buyer module. This is further used in determining packet urgency and the significance of the users in cloud gaming systems. A well-developed adaptive segment request strategy proposed by (He et al.; 2014) with respect to client buyer and MAC queue..

This paper Chuah et al. (2014) puts forward details involving the evolution of cloud gaming into an environment sustainable in reducing hardware utilization and improving the software utilization. With reduced game streaming bit rate, graphic processing ability is improved significantly in mobile gaming. This provides with high scalability, long life for hardware accessories, and cost reduction.

The technology growth in the field of cloud computing has contributed many opportunities to various industries and one such is Gaming. Cloud gaming allows users to access high quality games remotely through the internet irrespective to the devices configuration. This technology cuts down the barrier of compatibility issue while accessing certain high-quality games from devices that has lower computational power. According to Shea et al. (2013) Cloud Computings features such as on demand resource along with offloading techniques, these cloud gaming applications came into existence and were largely benefitted. Having said, cloud gaming technology has entered into the field of cloud computing and is exploring many ways to deepen its root. After careful observation on the state of art in cloud gaming platform, the author describes about how the design pattern for cloud gaming is derived. The major factors considered for the design were delay tolerance and

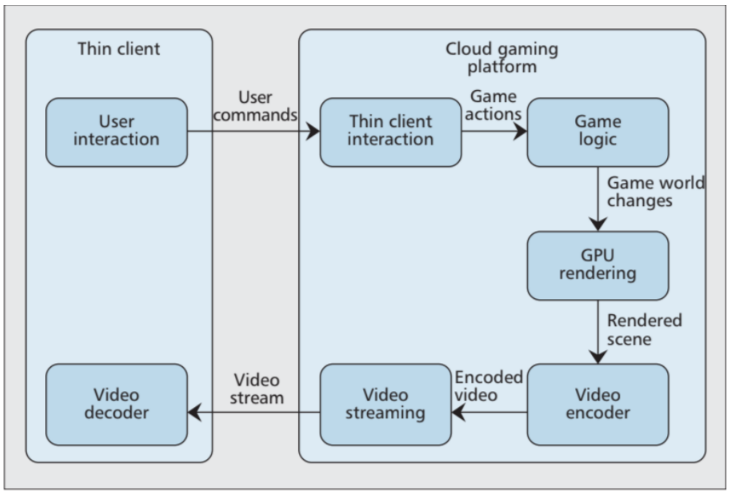

Figure 2: Architecture For cloud gaming

streaming seamlessly and encoding the video. Since these two were critical in providing fulfilled gaming experience, developers designed the cloud gaming framework based on this. For analyzing the interaction delay the author has configured a demo test system and has installed into the server. To analyze its performance and various other metrics corresponding to GPU usage, he used a tuning software called MSI Aferburner Shea et al. (2013). These tests were carried out to measure three major factors: Performance Optimization, Network latency, Image Quality. Apart from all these, according to Shea et al. (2013) there has been constant improvement from hardware side as well toward the cloud gaming platform. One such is NVIDIAs GeForce grid graphical processor. In a concluding note, this cloud gaming has been evolved to a certain extent that it facilitates all the advance opportunity to experience gaming in almost all the devices from lower configuration to high configuration. These tests provide positive results and opens many challenging possibilities to enhance its operational abilities.

The author (Yadav et al.; 2018) introduced a novel flexible solution for the users in the cloud gaming platform by providing a Gaming as a Service. They found an alternative way in reducing cost, by increasing the anticipated factors for the gaming users to provide a valuable gaming experience. The novel approach of component based game was implemented to reduce the resource management and throughput optimization in the cloud platform. The approach significantly increases the Quality of Services for the game players by efficiently reducing the network data in the terminal of the users.

3 Methodology

The solution which is proposed here is totally based on enhancing the performance of the cloud gaming system. To implement the cloud gaming system. Hence the Docker CE is used for creating the containers, the container must be created in the docker image. The docker image will be build with ubuntu OS in it with the required specifications, where it will be having the prerequisites files in the image. To allocate the graphics resources, NVIDIA graphic driver is used in the containers. The prerequisites software which is important to run the cloud gaming system such as GNU c++ compiler, package config files and with a list of library files with binaries and developers file which will be installed in the

containers.
The source code will be executed inside the containers. The output and input data between the client and the server is executed based on the (Huang et al.; 2014), by which these inputs and outputs are transmitted into video and audio format where the video will be encoded by capturing the instance which runs in the server side. It works based on screen capturing in real time and transmitted to the client where the screen rendering is based on the real time. It follows 2 implementation formats that is desktop capture module and API interception module. By video streaming the FPS rate can be maintained. For the Audio streaming ALSA library is used where it captures the audio and transmits to the user by audio encoding using the library. The containers will be connected with client through a x-window called VNC. By which it will accessed through the console. From the container the game will be deployed to the client side by accessing the configurations file based on the events which is selected from the application. After hosting the system, the benchmark tests will be initialized using Phoronix - test suite. This testing process will be initialled based performance of the overall system based on the FPS of the game. Where it will be tested on the game directly and on the cloud gaming system.

4 Design Specification

4.1 Architecture for the base cloud gaming system

The user initially logs onto the system through a portal server that shows a list of games that are available. User selects the game that he/she prefers and sends out a request to play it. On receiving the request from the user, the portal searches and finds the available server for that particular game. It launches the game on the server and returns the URL of that server to the user. The user logs on to the server and plays the game. If the login and game selection are both sent from a client, it needs not, a user interface.

The two types of network flows present in the architecture are the control flow and the data flow. The data flow is used for audio and video frame streaming from the server to client. The control flows are used for delivering user actions from the client to the server. The architecture of cloud gaming system allows it to handle both PC based and Web-based games was explained by (Huang et al.; 2014) . The game selected by the user runs on a game server and an agent runs along with the game on the same server, this agent can be a stand-alone process or can be a thread attached into the game selected. The choice between the two depends on the manner of implementation of the game and its type. The agent has two fundamental tasks, one is which, the A/V frames of the game need to be captured and encoding frames to the client using the data flow, the second task being, interaction of the agent with the game. On receiving the data from the users actions from the client, the agent must behave as the user and play along the game by re-playing all of the users gestures. Albeit, there does not exist a standard protocol for involving the users actions. For this reason, we manually select the framework and implement the transport rules for user actions (Huang et al.; 2014).

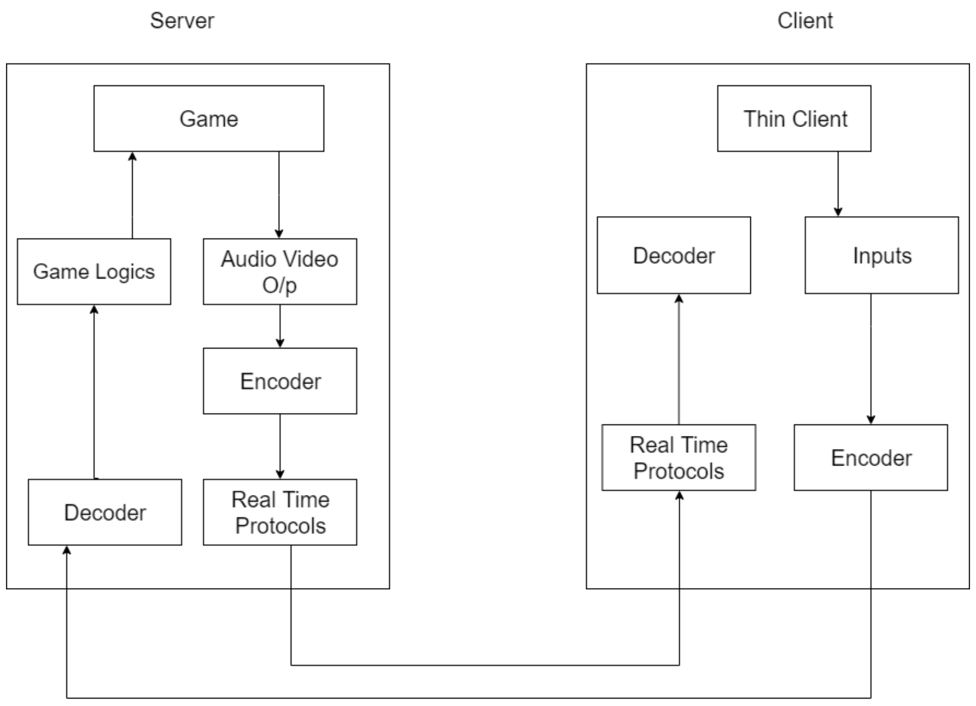

Figure 3: GamingAnywhere Architecture

A The client is a game console created in a custom manner and is implemented by combining the RTSP/RTP multimedia player and a keyboard/mouse logger. The system architecture of cloud gaming system, traditionally permits the client as the server provides encoded A/V frames using the standard RTSP and RTP protocols based on (Huang et al.; 2014). In this manner, an observer may be allowed in watching a game play by just accessing the particular game URL with the complete featured multimedia players. These multimedia players may include VLC media player, that are available locally on most of the systems.

4.2 Architecture for the cloud gaming system on top of the container

Containers is used for os level virtualization by which multiple users can access the kernel. It will handle the instance mare scalable and reliable, n number of instances can be implemented in it. It reduces the cost and resource usage. Basically, containers dont need a huge OS to run it. The containers can directly communicate with the hardware. The containers will be placed on the host OS. Here the containers are implemented through Docker image. The cloud gaming application is place inside the containers. The containers will be hosting the cloud gaming system. Hence the cloud gaming inside the container will be considered as the server where else from the container the client will be connected through X - window. Thus, the X-window will help the user to access the application from the container.

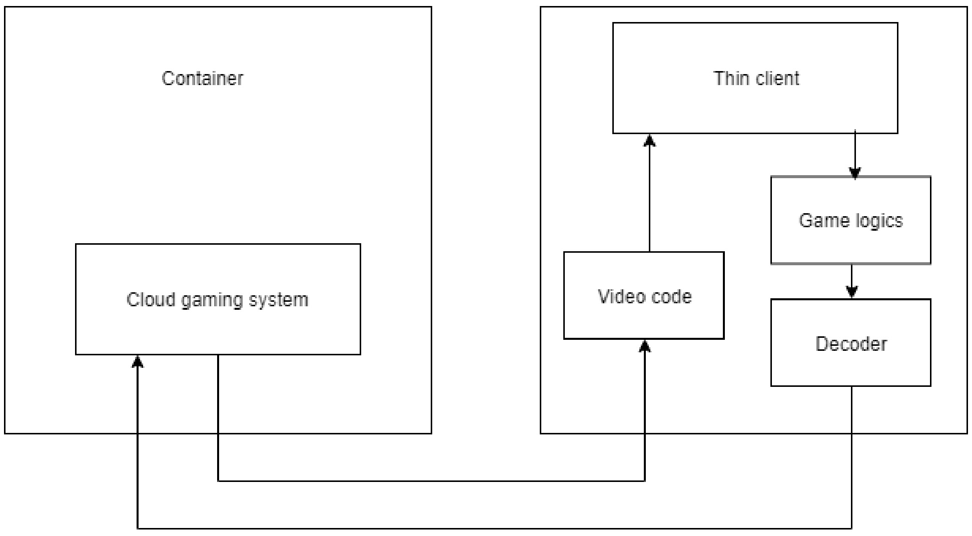

Figure 4: Architecture for the Cloud gaming system with container

5 Implementation

The proposed solution is by developing the cloud gaming system with the help of existing open source cloud gaming system called GamingAnwhere (Huang et al.; 2014). Thus, system will be implemented inside the containers using a docker image. The current cloud gaming system will virtualize their GPU setup through hypervisor based on the priority of the user. Here we are Developing the cloud gaming system on the docker container system

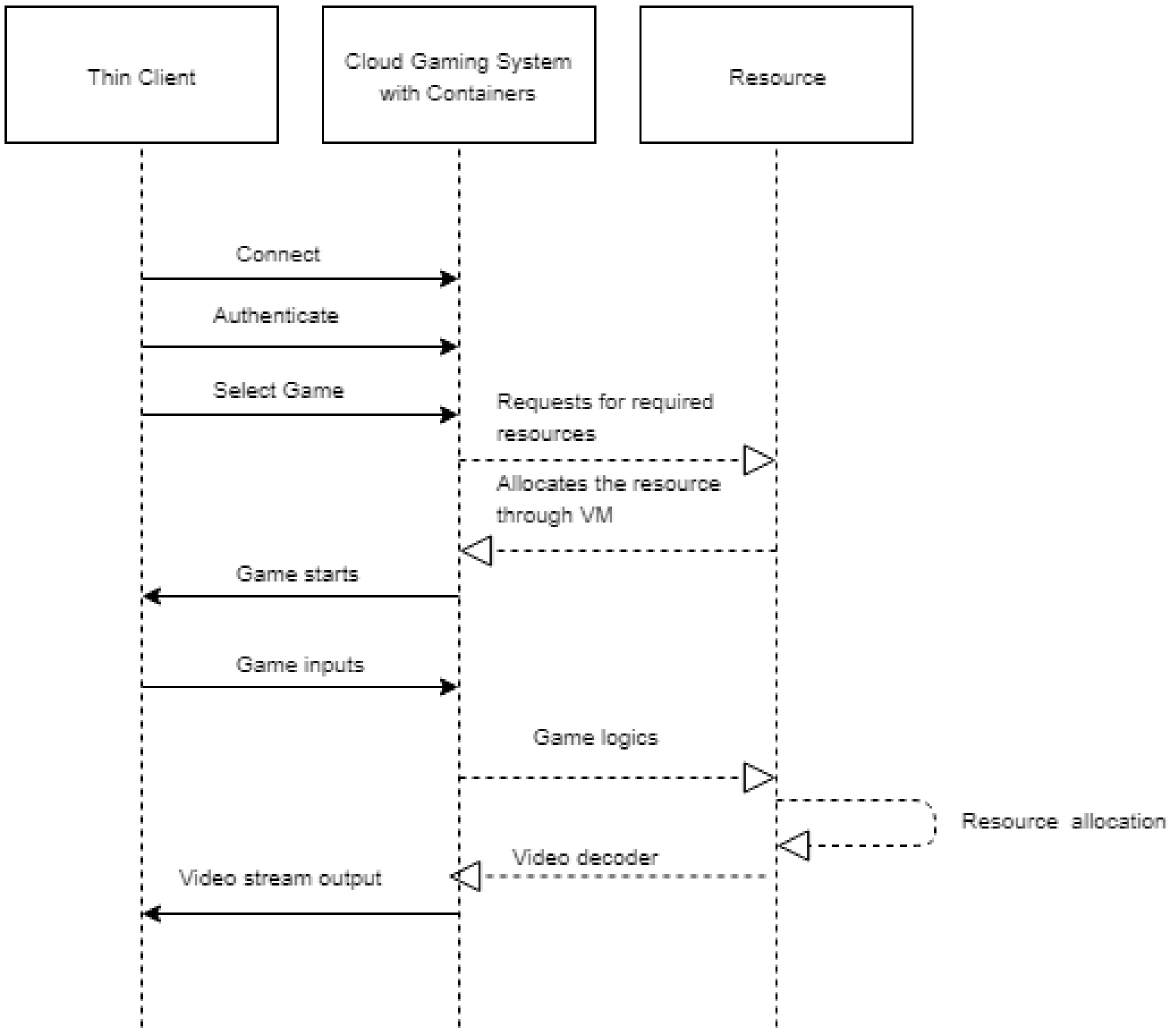

Figure 5: The flow Diagram of Interaction between the Thin client, Cloud gaming system and the Resource

5.1 Cloud gaming system

This cloud gaming system is initialized with help of open source cloud gaming system called gaming anywhere. This system was developed based on client server process, where it follows 2 process namely periodic mode and event-driven mode. In the periodic driven the whole host system will be virtualized to the client side while in the other hand in the event-driven mode the particular application will only be virtualized. The application is developed using C++ by which the application which is implemented on Linux OS is totally depends on number of library files like libbz2, libfreetype6, libva1, libasound2, libsdl2, zlib1g, libswscale2, libX11, libstdc++6 and libxtst6 hence it is consists of development file and binary file. These are the library files which is used for video processing, audio processing, file compressing and data compressing.

- libbz2 - It is used for the package container which will be used for the data compressor.
- libfreetype6 - It is used for managing font style, anti-aliasing and performance.
- libva1 - It is a video acceleration API which is used for GPU acceleration in Linux.
- libasound2 - This lib file is architecture for the sound management in the ALSA library.
- libswscale2 - this lib is mainly used for video scaling for the rendered video.
- libx11 - it is used in X window which supports the protocols of the client lib in the C++
- libstdc++ - this is a library package which is used to built the programs within the GNU compiler
- libxtst6 - it is used as the client interface for x window protocol.

5.2 Initializing Containers

The containers are being initialized by using Docker. So, to execute the containers, Docker-ce is installed. After the installation we need to permit the local connection to the X server by accessing the root file by this process it will allow to execute the graphics application run inside the docker containers. Each and every container which is created will be having its own container ID where else we can pull the docker image from the docker website which ever docker container we need. After creating the container, we can access the container through container image. Here we are creating a docker container with ubuntu installed which is customized based on our requirements. To create a docker image, an installation file is created which consists of software, driver installation and updates which helps to build a docker image. An interactive file is created where it is used for calling the docker image. Nvidia driver is installed, which used for the graphic driver settings. The CRIU functionality is used for the live migration which helps to provide the docker with the checkpoint and the restore point. Hence it helps to create a backup point, if the application fails or throws an error, the backup point will help to start from that part. The X-server should be started for the containers to run with graphic application. After initializing the backup point. To create a docker image, the cloud gaming system is installed inside the docker and with the required library files and the execution file for installing the cloud gaming system. The docker image with the cloud gaming system will be created. After building the docker image create the container with image name in the required location.

6 Evaluation

This cloud gaming application is developed based on the increase in the overall performance of the system by calculating the FPS rate of the game running in the in the cloud gaming system and the CPU usage as well. These FPS rate will determine the quality, when there is a higher FPS rate the quality of the game will be high. The most important process of the cloud gaming is rendering the video from server to client using the Server GPU driver. When the game runs in the system directly there will be higher rate of FPS, The GPU rendering will be high and overall performance will be high. Here we are going to compare the performance of the game by its FPS rate and over all performance. To measure these performances with Phoronix test suite benchmark tool. By taking these results the FPS rate and the overall performance will be determined. These results will be based on the cloud gaming system which is implemented inside the containers. these benchmark results will be compared with results of the previously tested results of GamingAnywhere application and running the game on a PC. The specification of the host system which runs the Cloud gaming system with containers has a 8GB of ram, NVIDIA GPU driver and i7 7th gen processor. Here the docker will be installed in the ubuntu OS and connected to the client

6.1 Test case 1

Here the benchmarking test is executed to analyse the overall performance for the game where we will be running an open source game called AssaultCube. The game is first run on the system and the results will be derived from the Phoronix test suite benchmark tool

Hence by above results was derived from the benchmarking tool where the overall performance of the application is analyzed based on the FPS(frames per second) rate of the each game. The performance of the game in the PC is maximum. While the scores for the game when it is hosted from the Gaming anywhere application is much lower, but when the similar system is initialized in the docker,Where the performance of the docker image which contains the cloud gaming system is quite similar to the performance of the game in PC . Hence test case 1 proves that the cloud gaming system inside the docker has a better performance this test case is valuated based on the FPS rate of the game. The performance results are shown in Figure 5.

System with AssultCube	Docker	Gaming anywhere	PC
Benchmark score	164	141	168

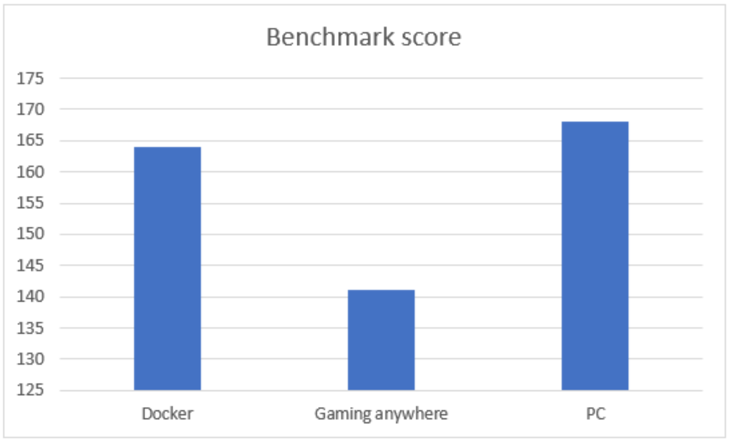

Figure 6: Test Case 1 compared based on FPS rate

6.2 Test Case 2

In this test case Cube2: Sauerbraten game is executed by which the overall performance of the game will be analyzed using the same benchmark tool based on the FPS rate (Phoronix test suite).

System with Sauerbraten	Docker	Gaming anywhere	PC
Benchmark score	178	152	185

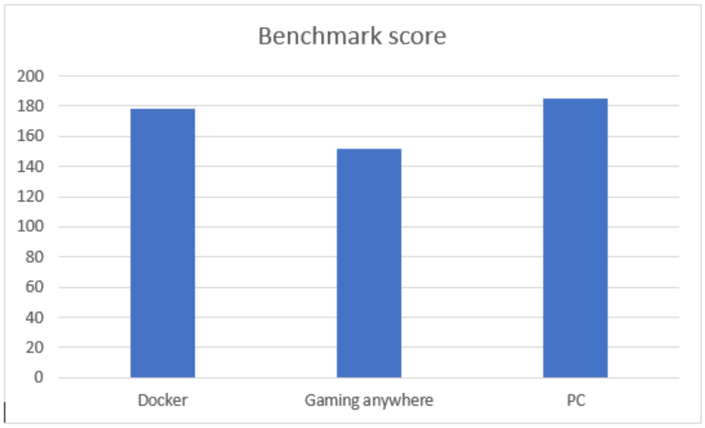

Figure 7: Test Case 2 compared based on FPS rate

From the above results of the benchmark shows that the overall performance the Gaming anywhere system is increased when the similar system is implemented in the containers. the results are shown in Figure 6

6.3 Discussion

From the above 2 cases the games are running with different graphics settings, where else the games are AssultCube and Cube2: Sauerbraten. When the games run directly from the Gaming anywhere there is considerable amount of performance drop in the games. When the similar cloud gaming system is initialised through Docker containers aby pulling the docker image. The overall performance increases. By which the docker system has made the system more performable, reliable and scalable.

7 Conclusion and Future Work

In this paper the proposed solution increases the overall performance of the cloud gaming system which is initialized in Docker container, which helped to increase the FPS rate and overall performance of the cloud gaming system.

In future the cloud gaming system will be enhanced with the auto-scaling process in the docker and with Advanced Scheduling algorithm the handle the resource more efficiently and by this many high graphics game can be run in the Cloud gaming system which would be more reliable and scalable.

References

Cai, W., Chi, Y., Zhou, C., Zhu, C. and Leung, V. C. M. (2018). Ubcgaming: Ubiquitous cloud gaming system, *IEEE Systems Journal* pp. 1–12.

Chen, K., Huang, C. and Hsu, C. (2014). Cloud gaming onward: research opportunities and outlook, *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–4.

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

He, L., Liu, G. and Yuchen, C. (2014). Buffer status and content aware scheduling scheme for cloud gaming based on video streaming, *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–6.

Hong, H., Chen, D., Huang, C., Chen, K. and Hsu, C. (2015). Placing virtual machines to optimize cloud gaming experience, *IEEE Transactions on Cloud Computing* **3**(1): 42–53.

Huang, C.-Y., Chen, D.-Y., Hsu, C.-H. and Chen, K.-T. (2013). GamingAnywhere: An open-source cloud gaming testbed, *Proceedings of ACM Multimedia 2013 (Open Source Software Competition Track)*.

Huang, C.-Y., Chen, K.-T., Chen, D.-Y., Hsu, H.-J. and Hsu, C.-H. (2014). Gaminganywhere: The first open source cloud gaming system, *ACM Trans. Multimedia Comput. Commun. Appl.* **10**(1s): 10:1–10:25.
URL: *http://doi.acm.org/10.1145/2537855*

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

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 Trans. Archit. Code Optim.* **11**(2): 17:1–17:25. CORE RANKING:A.
URL: *http://doi.acm.org/10.1145/2632216*

SanWariya, A., Nair, R. and Shiwani, S. (2016). Analyzing processing overhead of type-0 hypervisor for cloud gaming, *2016 International Conference on Advances in Computing, Communication, Automation (ICACCA) (Spring)*, pp. 1–5.

Seneviratne, C. and Leung, H. (2011). A game theoretic approach for resource allocation in cognitive wireless sensor networks, *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1992–1997.

Shea, R., Fu, D. and Liu, J. (2015). Cloud gaming: Understanding the support from advanced virtualization and hardware, *IEEE Transactions on Circuits and Systems for Video Technology* **25**(12): 2026–2037.

Shea, R., Liu, J., Ngai, E. C. . and Cui, Y. (2013). Cloud gaming: architecture and performance, *IEEE Network* **27**(4): 16–21.

Usman, M., Iqbal, A. and Kiran, M. (2017). A bandwidth friendly architecture for cloud gaming, *2017 International Conference on Information Networking (ICOIN)*, pp. 179–184. CORE RANKING:B.

Wang, B., Ma, R., Qi, Z., Yao, J. and Guan, H. (2016). A user mode cpu-gpu scheduling framework for hybrid workloads, *Future Gener. Comput. Syst.* **63**(C): 25–36. CORE RANKING:A.
URL: *http://dx.doi.org/10.1016/j.future.2016.03.011*

Wang, Z., Xu, W., Yang, J. and Peng, J. (2009). A game theoretic approach for resource allocation based on ant colony optimization in emergency management, *2009 International Conference on Information Engineering and Computer Science*, pp. 1–4.

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

Yadav, R. R., Sousa, E. T. G. and Callou, G. R. A. (2018). Performance comparison between virtual machines and docker containers, *IEEE Latin America Transactions* **16**(8): 2282–2288.

Zhang, C., Yao, J., Qi, Z., Yu, M. and Guan, H. (2014). vgasa: Adaptive scheduling algorithm of virtualized gpu resource in cloud gaming, *IEEE Transactions on Parallel and Distributed Systems* **25**(11): 3036–3045. CORE RANKING:A*.

Zhang, L. and he Zhou, J. (2017). Task scheduling and resource allocation algorithm in cloud computing system based on non-cooperative game, *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 254–259.

Zhang, W., Liao, X., Li, P., Jin, H., Lin, L. and Zhou, B. B. (2018). Fine-grained scheduling in cloud gaming on heterogeneous cpu-gpu clusters, *IEEE Network* **32**(1): 172–178.

Zhang, Y., Qu, P., Cihang, J. and Zheng, W. (2016). A cloud gaming system based on user-level virtualization and its resource scheduling, *IEEE Transactions on Parallel and Distributed Systems* **27**(5): 1239–1252. CORE RANKING:A*.