Improving Quality of Service Metrics and User Perception in VR-based Cloud Gaming

MSc Research Project
Cloud Computing

Muhammad Muhteshim Ghazali
Student ID: 22190228

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Muhammad Muhteshim Ghazali
Student ID:	22190228
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	12/08/2024
Project Title:	Improving Quality of Service Metrics and User Perception in VR-based Cloud Gaming
Word Count:	8,177
Page Count:	21

Signature:	
Date:	14th September 2024

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.	☐

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

Improving Quality of Service Metrics and User Perception in VR-based Cloud Gaming

Muhammad Muhteshim Ghazali
22190228

Abstract

Exploring how cloud gaming services can be integrated with virtual reality (VR) technology, with an emphasis on improving Quality of Service (QoS) metrics including latency. VR head mounted displays (HMDs) have advanced over the last ten years, opening up immersive experiences across a range of industries. In order to provide wider accessibility, the study divides VR HMDs into three categories: mobile, standalone, and tethered devices. It emphasizes the significance of standalone devices with integrated computing capability. Latency and bandwidth needs are two obstacles that stand in the way of VR and cloud gaming convergence, as demonstrated by services like Google Stadia and Nvidia's GeForce Now. In order to mitigate these problems and ensure fluid and responsive gaming, technologies like edge computing and 5G connectivity are essential. This study indicates that Gradient Boosting Machines (GBM) can forecast network latency better than Random Forest and AdaBoost. This is significant since it demonstrates how theoretical models may be implemented in practical settings. The study analyzes dynamic resource provisioning in a cloud gaming environment using CloudSim, a cloud simulation tool. The simulations enable scalable and effective resource management by simulating real-world cloud infrastructures. The findings highlight how important cutting-edge machine learning models and cloud computing infrastructure are to improving VR gamer experiences. This research advances our knowledge of how cloud gaming services are optimized and offers predictions for the advancement of virtual reality and interactive entertainment in the future.

1 Introduction

Virtual Reality (VR) technology has rapidly evolved over the past decade, particularly since 2012 when significant corporate funding boosted research. The last five years have seen the release of numerous consumer and professional-grade stereoscopic head-mounted displays (HMDs), integrating VR into scientific and sensory applications. VR HMDs are broadly classified into three categories: mobile, standalone, or tethered devices. Standalone devices have built-in processing power, while tethered devices require external computing hardware. Mobile headsets utilizing smartphones enable users to access VR content, expanding applications into consumer studies, therapy, and gaming Wang et al. (2021).

The rise of virtual reality (VR) gaming has been remarkable, revolutionizing interactive entertainment. VR gaming uniquely immerses players in digitally generated environments. As demand for immersive experiences grows, integrating cloud gaming services has become crucial. This technology meets the processing demands of complex virtual environments, allowing users to overcome local hardware limitations and potentially broadening VR's audience significantly Dani (2019). Integrating VR with cloud gaming presents challenges, especially with Quality of Service (QoS) metrics like latency and resolution. Balancing low latency and high resolution is crucial to maintaining an immersive VR experience, as even slight delays can diminish user immersion. Technologies like edge computing are being developed to address these challenges by processing data closer to users. Edge computing deploys servers at the network edge, reducing the time for data to travel, which is critical for VR applications sensitive to delays.

Google's Stadia and NVIDIA's Ge-Force Now exemplify effective use of edge computing to deliver high-quality gaming experiences with reduced latency. These platforms support seamless game-play with high-resolution graphics, showcasing the potential of edge computing to enhance VR gaming. Another development that holds promise for improving QoS in VR cloud gaming is the advent of 5G connectivity. With its high data transfer speeds and low latency, 5G has the potential to revolutionize the delivery of VR content. For instance, Verizon's 5G network has been tested to support cloud gaming services with impressive results, offering faster response times and smoother gameplay. This improvement is particularly crucial for VR gaming, where the speed of data transfer can make or break the immersive experience Zhao et al. (2021).

1.1 Research Question

How do Quality of Service (QoS) metric: latency impacts the immersive nature of Virtual Reality (VR) gaming experiences within a cloud gaming environment, and what unique challenges and opportunities do these metrics present for VR-based cloud gaming?

1.2 Research Objectives and Contributions

As virtual reality gaming continues to gain popularity, it is vital to meet consumer expectations for a seamless and immersive experience. Understanding the importance of QoS measures ensures that cloud gaming services align with user expectations, resulting in higher levels of satisfaction Deng et al. (2023). This research problem is novel as it examines the intricate relationship between the immersive quality of VR gaming experiences and QoS metrics in a cloud gaming environment. The implications extend to the gaming industry and the overall user experience. The research aims to address several crucial factors: the increasing importance of cloud gaming services in meeting VR gaming's computational demands, the need to balance low latency and high resolution, the potential of edge computing technologies, and the integration of 5G connectivity to address latency issues. By critically analyzing a user-centric approach and considering the impact of QoS metrics on the overall user experience in VR gaming, the research seeks to identify challenges and opportunities. Ultimately, the goal is to offer guidance for future developments, innovations, and enhancements in VR cloud gaming services.

The fusion of VR technology and cloud gaming represents a significant advancement in interactive entertainment. Addressing the technical challenges related to QoS metrics, such as latency and resolution, through innovative solutions like edge computing and 5G connectivity, can significantly enhance the user experience. As this field continues to evolve, ongoing research and development will be essential to unlocking the full potential of VR in cloud gaming, providing more immersive, responsive, and satisfying experiences for users around the world.

2 Related Work

The convergence of advanced machine learning algorithms, immersive virtual reality (VR) experiences, and cloud gaming has expanded computational capabilities and user experiences. Understanding the performance of Gradient Boosting Machines (GBM) relative to other algorithms is crucial for optimizing these technologies. This section explores GBM's effectiveness compared to other algorithms, highlighting strengths and weaknesses. Integrating cloud gaming and VR requires scalable machine learning solutions to ensure seamless experiences, emphasizing the balance between technical performance and user satisfaction. This discussion offers a comprehensive review of GBM and its application in this evolving technological landscape.

2.1 Integration of Cloud Gaming and Virtual Reality

The integration of cloud gaming and VR has transformative potential in the gaming industry, offering unprecedented immersion and accessibility without the need for expensive hardware. However, challenges such as latency, bandwidth requirements, and network stability persist, impacting seamless gameplay.

Xu et al. (2018) introduced Transparent Gaming (TG) cloud, a method enabling users to play high-end desktop games over the Internet efficiently. Using TG-SHARE technology, it optimizes bandwidth usage by leveraging consumer GPUs and adaptive compression based on gameplay and network conditions. This approach simplifies access to high-quality gaming experiences from any Internet-connected location. Ethical considerations arise with the combination of cloud gaming and VR, including data security, privacy, and digital rights management. Concerns about user data collection for personalized interactions and advertising, alongside issues of addiction and screen time, underscore the need for ethical gaming practices and regulatory oversight in this evolving technological landscape.

2.2 User-Centric Approach & Technological Advancement

Efficient network resource management is increasingly critical for cloud-based Virtual Reality (VR) games to optimize user experiences. Anticipating traffic patterns, especially frame sizes, presents challenges that Machine Learning (ML) and Deep Learning (DL) models aim to address. Vaidya et al. (2023) use real-world data from a cloud-based VR gaming testbed to compare prediction accuracy of DL models like Long Short Term Memory (LSTM) and Convolutional Neural Networks (CNNs) against traditional ML methods. DL models achieve a 30% increase in accuracy, with Transfer Learning (TL) further enhancing predictions by up to 54% in dynamic environments.

This research establishes foundational insights into VR traffic patterns, offering recommendations for future studies to improve prediction algorithms with TL and analyze diverse VR traffic scenarios. It emphasizes TL's role in reliable network resource management for cloud-based VR gaming, crucial for enhancing prediction accuracy and system performance.

Sukhmani et al. (2019) discuss how edge computing and caching can address latency and bandwidth challenges in AR/VR applications and the tactile internet. They highlight the potential of 5G networks, edge caching, and mobile edge computing to support real-time interaction and haptic feedback, essential for immersive user experiences across entertainment, healthcare, and robotics. These technologies bring content and computing closer to users, improving responsiveness in interactive VR environments.

2.3 Balancing Latency

In machine learning, Gradient Boosting Machines (GBM) have gained attention for their robust performance across predictive tasks. Comparing GBM with other key algorithms, Random Forest and AdaBoost; reveals insights into their strengths and weaknesses. This section offers a detailed comparison, emphasizing why GBM is often preferred for its superior performance in diverse applications.

2.3.1 Random Forest

Random Forest, introduced by Breiman (2001), is an ensemble learning method that builds multiple decision trees during training. It aggregates predictions by voting (for classification) or averaging (for regression), known for its simplicity and robustness in handling large datasets to mitigate overfitting. However, it can become computationally demanding with larger datasets and lacks interpretability compared to individual decision trees.

In contrast, Gradient Boosting Machines (GBM) also use decision trees but sequentially build them to correct errors of previous trees. This iterative refinement often results in higher predictive accuracy than Random Forest, as demonstrated in studies like Natekin and Knoll (2013). However, GBM requires careful hyperparameter tuning and entails higher computational complexity due to its sequential nature.

While both Random Forest and GBM leverage ensemble learning with decision trees, they differ fundamentally in how they construct and refine their models. Random Forest benefits from simplicity and reduced risk of overfitting, while GBM excels in predictive accuracy at the cost of increased computational resources and tuning efforts. GBM is often preferred in applications prioritizing accuracy despite these challenges.

2.3.2 Adaptive Boosting (AdaBoost)

Adaptive Boosting (AdaBoost), developed by Freund et al. (1996), is another learning technique that enhances weak classifiers by focusing on misclassified instances iteratively. It effectively transforms weak learners into strong classifiers, especially when using simple base classifiers like decision stumps. AdaBoost's strength lies in its simplicity and effectiveness in boosting performance. However, it can be sensitive to noisy data and may overfit when weak classifiers are overly complex.

According to Bahad and Saxena (2020), when Comparing AdaBoost with Gradient Boosting Machines (GBM), both employ boosting but differ significantly in approach. AdaBoost adjusts instance weights for misclassifications, whereas GBM optimizes a differentiable loss function, offering greater flexibility and power in handling complex datasets. GBM's ability to minimize diverse loss functions tailored to specific tasks enhances its performance in machine learning competitions and real-world applications, demonstrating superior adaptability and precision.

Moreover, as highlighted by Taieb and Hyndman (2014),GBM's superior performance in machine learning competitions, including those on platforms like Kaggle, underscores its adaptability and precision. Compared to AdaBoost, which is effective and straightforward, GBM's ability to optimize various loss functions and handle complex data structures positions it as a more powerful tool in practical applications. This capability to manage diverse and challenging datasets solidifies GBM's reputation as a superior choice in predictive modeling scenarios.

3 Methodology

In this research, I aimed to optimize cloud gaming environments by simulating dynamic resource provisioning and latency management using CloudSim, mentioned by Shahid et al. (2023). The process involved several key steps, including data acquisition and preprocessing, algorithm selection and evaluation, and the simulation of cloud environments with CloudSim.

Initially, I obtained a dataset comprising network packet captures that provided detailed information on gaming traffic. Using editcap, I segmented the dataset into files containing approximately 15,000 to 20,000 packets each, which allowed for more manageable data processing and analysis. Next, I converted the packet capture files (pcapng) to CSV format. The CSV files included essential packet information such as source IP, destination IP, time, protocol, packet length, and additional metadata as extensively explained in section 2 The modification made employing data processing tools for data manipulation and analysis easier. In order to provide high-quality inputs for machine learning models, data preprocessing entailed cleaning and organizing the data. This involved choosing important features for model training, such as packet length, protocol type, and time, and managing missing values as well as normalizing packet lengths and other pertinent metrics.

I identified three machine learning algorithms AdaBoost, Gradient Boosting Machines (GBM), and Random Forest to forecast network latency using the cleaned dataset. The most effective model for forecasting latency in cloud gaming scenarios was identified by evaluating each algorithm using a range of performance parameters, such as accuracy, precision, recall, and F1-score. GBM performed better across all evaluation measures, therefore after careful comparison, I went with it. The perfect option was GBM because of its resilience to overfitting and capacity to manage intricate, non-linear interactions. Using CloudSim to simulate resource provisioning, the expected latency values from GBM were utilized.

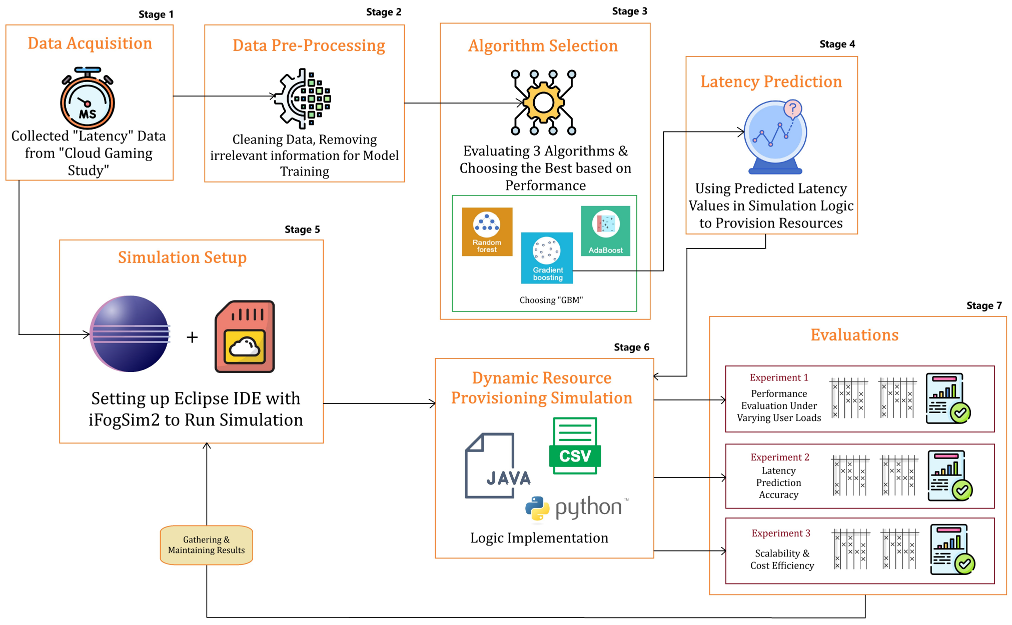
Stage 1
Data Acquisition
MS
Collected "Latency" Data from "Cloud Gaming Study"
Stage 2
Data Pre-Processing
Cleaning Data, Removing irrelevant information for Model Training
Stage 3
Algorithm Selection
Evaluating 3 Algorithms & Choosing the Best based on Performance
Random forest
Gradient boosting
AdaBoost
Choosing "GBM"
Stage 4
Latency Prediction
Using Predicted Latency Values in Simulation Logic to Provision Resources
Stage 5
Simulation Setup
+
Setting up Eclipse IDE with iFogSim2 to Run Simulation
Gathering & Maintaining Results
Stage 6
Dynamic Resource Provisioning Simulation
CSV
JAVA
python
Logic Implementation
Stage 7
Evaluations
Experiment 1
Performance Evaluation Under Varying User Loads
Experiment 2
Latency Prediction Accuracy
Experiment 3
Scalability & Cost Efficiency

Figure 1: Different Stages of Complete Workflow

Using CloudSim, I was able to replicate the behavior of real world cloud services such as AWS Wavelength. We were able to simulate a dynamic cloud system using CloudSim, which modifies resource allocation according to anticipated latency and user load. Datacenters that represented cloud servers were configured in the simulation, with settings for memory, bandwidth, and computing power that reflected actual cloud environments.

Virtual machines (VMs) and cloudlets (tasks) were created to represent gaming sessions, simulating user interactions and workload in the cloud environment. Zhu et al. (2022) worked and I am doing the same, the network topology was configured using NetworkTopology in CloudSim, incorporating latency values derived from the GBM model to simulate network conditions accurately. The simulation also incorporated a mechanism to dynamically provision resources based on the number of active users and the predicted latency. As user load increased, additional VMs were provisioned, and resources were scaled down as the load decreased, ensuring optimal performance and latency.

The CloudSim simulation provided insight on how dynamic resource adjustments based on current network circumstances might optimize the cloud gaming environment. The goal of this strategy was to improve user experience while preserving low latency and effective resource usage.

As depicted in Figure 1, the methodology combined data preprocessing, machine learning, and simulation techniques to address the challenges of optimizing cloud gaming environments. By accurately predicting latency and simulating resource management in CloudSim, I demonstrated a viable solution for enhancing cloud gaming performance, contributing to the development of more responsive and scalable cloud gaming infrastructures.

I decided to use an Amazon EC2 instance for the machine learning model training and CloudSim simulations rather than a local computer to make sure my solution is scalable and ready for the cloud. Selecting from a range of instance types that are tailored for distinct workloads, enhanced computational capacity, and the freedom to scale resources dynamically based on demand are just a few benefits of running the simulations on an EC2 instance. I can more accurately simulate real-world cloud environments and make sure the system can manage the scaling needs inherent in cloud gaming scenarios by implementing my solution on an EC2 instance.

Choosing an instance type with enough memory and processing power to manage the demanding tasks of data preparation, model training, and simulation was part of setting up the EC2 instance. I decided to go with a general-purpose instance type, such the m5.large, which offers a good balance of network, CPU, and memory performance. Configuring the instance with the required software and libraries, including Java for running CloudSim, was part of the setup procedure. In order to meet the needs of the simulation and data transfer, I also made sure the instance had enough storage and network setups.

4 Design Specification

By combining VR technology with cloud gaming and employing innovative solutions such as edge computing and machine learning, this system aims to provide a seamless and immersive gaming experience while addressing the technical challenges related to QoS metrics.

4.1 System Architecture

The architecture of the proposed system integrates cloud gaming and VR technology, focusing on optimizing Quality of Service (QoS) metrics such as latency and resolution. The architecture includes the following components:

4.1.1 VR Head-Mounted Displays (HMDs):

Various types of HMDs (mobile, standalone, and tethered) are supported to ensure broad accessibility and usability.

4.1.2 Edge Computing Nodes:

Deployed near users to reduce latency by processing data closer to the user's location. These nodes handle initial data processing and forward the results to the central cloud servers.

4.1.3 Network Topology:

Configured to simulate real-world conditions using CloudSim, incorporating latency values derived from machine learning models to mimic network conditions accurately.

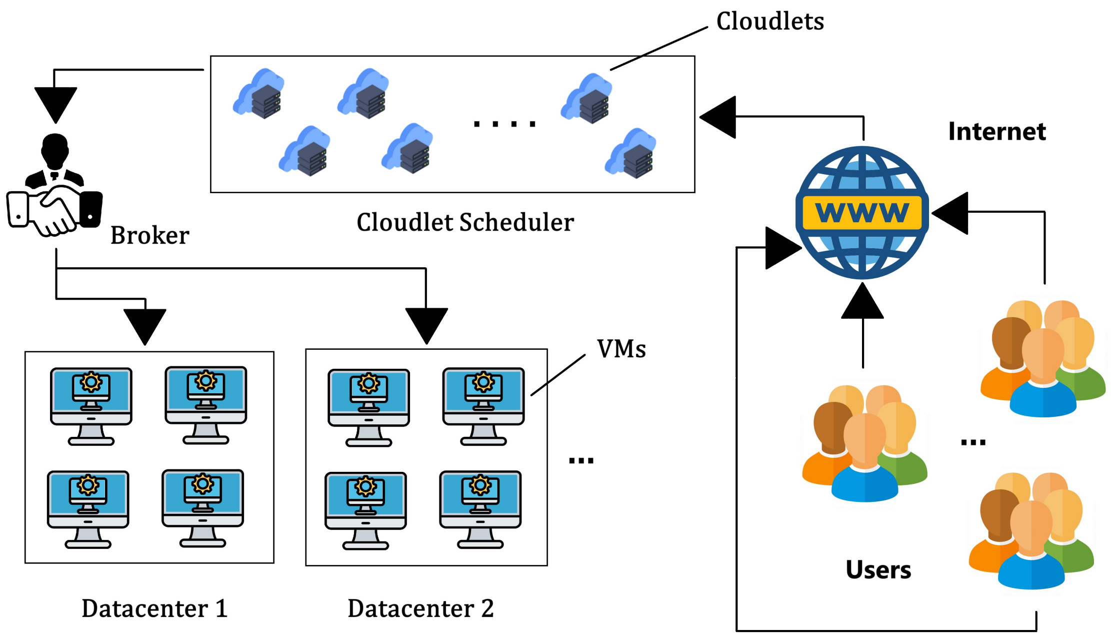

Figure 2: Basic Architecture of of System

4.1.4 Machine Learning Module:

Implements Gradient Boosting Machines (GBM) to predict network latency based on real-time data.

4.2 Data Processing and Machine Learning

4.2.1 Data Acquisition and Preprocessing:

Collect network traffic data from VR gaming sessions. Data includes source IP, destination IP, time, protocol, packet length, and metadata. Use editcap to segment packet capture files into manageable sizes (15,000 to 20,000 packets each) and convert them to CSV format for analysis. Handle missing values, normalize packet lengths, and select key features (packet length, protocol type, time). Extract relevant features for model training to improve prediction accuracy.

4.2.2 Machine Learning:

Evaluate AdaBoost, Gradient Boosting Machines (GBM), and Random Forest models based on performance metrics (accuracy, precision, recall, F1-score). Train models using preprocessed data and evaluate their performance. GBM is selected for its superior handling of non-linear relationships and robustness against over-fitting.

4.3 Simulation Environment

4.3.1 CloudSim Configuration:

Configure datacenters with parameters reflecting realistic cloud environments (processing power, memory, bandwidth). Create VMs and cloudlets to simulate gaming sessions, representing user interactions and workload. Implement using NetworkTopology in CloudSim, incorporating GBM-predicted latency values to simulate real-world conditions.

4.3.2 Dynamic Resource Provisioning:

Dynamically adjust resource allocation based on user load and predicted latency. Provision additional VMs as user load increases and scale down resources when the load decreases. Continuously monitor system performance and adjust resources to maintain optimal QoS metrics.

5 Implementation

In this section, I go over how the methodology's dynamic resource provisioning and latency management architecture is put into practice for cloud gaming environment optimization. Key components of the suggested solution are covered in each of the various crucial phases that make up the implementation process.

To start, a thorough preprocessing of the network packet capture data was done to provide a clean, organized dataset. The effectiveness of many machine learning techniques, such as AdaBoost, Gradient Boosting Machines (GBM), and Random Forest, in predicting network latency was then assessed. The most successful model was GBM, which used its prowess in managing intricate, non-linear interactions to precisely predict latency values. The resource provisioning algorithms in the CloudSim simulation environment were then based on these assumptions.

I reproduced the behavior of real world cloud services by establishing datacenters, virtual machines, and network topologies, and modeling dynamic modifications depending on real-time user loads and anticipated latencies. To ensure scalability and computational efficiency, which reflected real-world cloud conditions, the simulations were run on Amazon EC2. This section describes each stage in depth, including how CloudSim is configured, how machine learning models are run, and how dynamic resource management techniques are used to improve the performance of cloud gaming.

5.1 Data Loading and Processing

The objective of this study was to predict network latency using machine learning techniques, specifically through the implementation of a Gradient Boosting Regressor. The process began with the acquisition and preparation of network packet data, which was stored in a CSV file named merged_output_file_complete_lat20_data.csv. This dataset included key attributes of network packets such as source IP, destination IP, timestamp, and packet size, crucial for analyzing network performance.

The data was initially loaded into a Pandas DataFrame. This Python library offers powerful data manipulation capabilities, which were utilized to clean and transform the dataset. One of the primary preprocessing steps involved converting the ”frame.time” column from a string format with timezone information (”GMT Summer Time”) to a standardized datetime format. This was achieved by removing the timezone string and then parsing the datetime using the format %b %d, %Y %H: %M:%S. %f. This transformation was essential for accurate time series analysis and latency calculation.

Following the datetime conversion, the dataset was sorted by source IP, destination IP, and timestamp. Sorting was crucial for correctly computing the time intervals between packets, as network latency depends on the sequential order of packet transmission. The latency between consecutive packets was then calculated by grouping the data based on source and destination IP pairs and computing the time difference using ”diff()”. The resulting time differences, measured in seconds, were stored in a new column labeled ”latency”.

o ensure the quality of the dataset, rows containing NaN values in the ”latency” column were removed. These NaN values occurred for the first packet in each source-destination pair, where no preceding packet existed to calculate a time difference. The removal of these rows was necessary to avoid skewing the latency predictions.

5.2 Feature Selection and Data Encoding

With latency as the target variable (y), the remaining columns were designated as features (x). The feature set was refined by dropping non-informative columns such as frame.time, ip.src, and ip.dst, which were not used in the model training process. Since machine learning models require numerical input, categorical variables were converted into numeric format using one-hot encoding as used by Lean Yu and Lai (2022). This process transformed categorical feature values into a binary matrix, making them suitable for model training.

An 80/20 split was used to divide the dataset into training and testing sets. This separation made sure the model could be trained on a significant amount of the data and then tested on an unknown subset to see how well it performed. For latency prediction, the Gradient Boosting Machines was utilized due of its ability to handle intricate interactions with minimal overfitting. Through a sequential construction process, this algorithm creates an ensemble of decision trees, each of which tries to improve upon the mistakes of the one before it, increasing the overall accuracy of the model.

5.3 Dynamic Provisioning Simulation

As implemented in detailed by Le et al. (2022), but for a generic scenario, the main method initializes the CloudSim simulation environment by setting up the number of users, time, and tracing options. It creates two datacenters and a broker to manage virtual machines (VMs) and cloudlets. Initially, a VM and several cloudlets are added and submitted. The simulation starts and continues running until the specified simulation time is reached or until it is terminated. During each simulation step, it performs dynamic provisioning based on QoS metrics, adjusts resources based on user feedback, and periodically adds new cloudlets. Finally, it stops the simulation and prints the results of

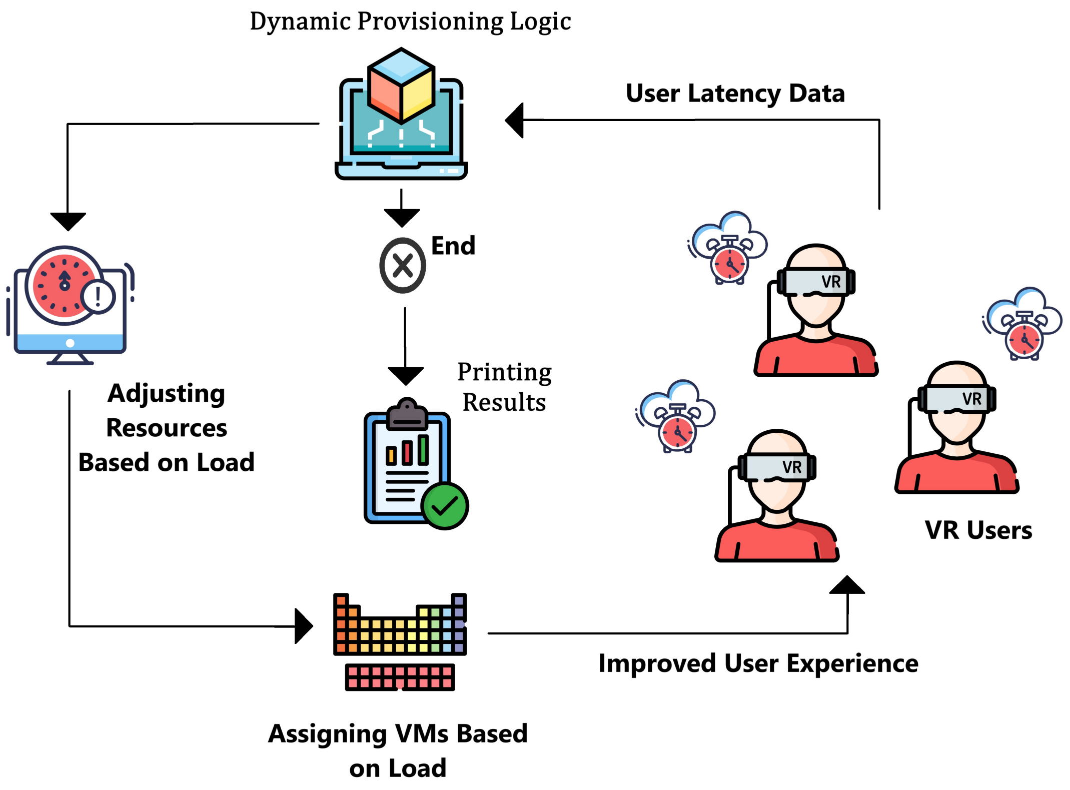
Dynamic Provisioning Logic
User Latency Data
End
Adjusting Resources Based on Load
Printing Results
VR
VR
VR
VR Users
Improved User Experience
Assigning VMs Based on Load

Figure 3: Dynamic Provisioning Logic Scenario

cloudlets and VM details.

5.3.1 Setting Up Simulation Resources

A datacenter is created and configured for the simulation using the "createDatacenter" method. A list of hosts is created, each having a designated capacity for RAM, bandwidth, storage, and computing power (MIPS). A host employs a time-shared virtual machine scheduler and has processor elements (PEs). The design, operating system, and budgetary constraints of a datacenter are specified. Subsequently, the procedure sets up the datacenter with these attributes, a basic virtual machine allocation policy, and no extra storage. In the event of an exception, the error is printed. The generated datacenter is then put back to use in the simulation.

A DatacenterBroker object is set up and returned by the "createBroker" method. Managing the submission of virtual machines and cloudlets to datacenters is the responsibility of a broker. "Broker" serves as its initial unique identifier. It prints the stack trace and returns null in the event that an error arises during startup. In order to guarantee that the broker is properly created for controlling the interaction between virtual machines and cloudlets in the datacenter environment, this technique encapsulates the process of constructing and handling the broker in the simulation.

A new virtual machine (VM) is created and added to the simulation using the "addNewVM" method. Based on the size of the VM list at the time, it assigns a unique ID and provides

VM specs like MIPS, RAM, storage, and bandwidth. The virtual machine (VM) utilizes a time-shared scheduler for cloudlets and is linked to a broker ID. Once the virtual machine (VM) has been created, it is appended to the ”vmList” and a notification is sent out along with the simulation clock time.

The ”addNewCloudlets” method creates and adds a new cloudlet to the simulation. It sets the cloudlet's ID, execution length, file sizes, and number of processing elements (PEs). It also assigns a full utilization model for CPU, memory, and bandwidth. Each cloudlet is associated with a broker ID and is assigned a ”QoSMetric” representing its latency, which is randomly generated between 10 and 40 milliseconds. The cloudlet is then added to the ”cloudletList”, and a message about the addition is printed with the current simulation clock time.

5.3.2 Building Logic for Resource Adjustment

In the cloud environment, virtual machine (VM) allocation is dynamically managed in large part via the ”adjustResourcesBasedOnLoad” method. Its main goal is to respond to latency estimates from a machine learning model by optimizing resource distribution. This technique modifies the number of virtual machines (VMs) in accordance with real-time latency projections, ensuring that the cloud system stays responsive and efficient.

Using the ”getPredictedLatencies” function, the technique first collects predicted latency data from an external Python script. This script predicts future latencies by using a Gradient Boosting Regressor model that was trained on previous network data. An array containing the expected delays for different cloudlet jobs or user interactions is returned as part of the latency estimates. The technique is able to make well informed decisions regarding resource adjustments by incorporating these forecasts.

Following the retrieval of predicted latencies, the method updates the Quality of Service (QoS) metrics associated with each cloudlet, a similar kind of experimentation is done by Selvaraj Kesavan and Vengatesan (2021). The qosMetricsMap parameter represents a mapping between cloudlet identifiers and their corresponding QoS metrics. Each QoS metric is updated with the predicted latency values, aligning the current QoS data with the latest forecasted information. This alignment is essential for accurate resource provisioning decisions.

The revised QoS measurements are then sorted by latency values in accordance with the technique. In order to prioritize tasks or users who are suffering higher latencies, sorting is done in ascending order. The process can identify which cloudlets or user interactions need more resources to reduce possible performance concerns thanks to this prioritization. The approach is able to precisely change the virtual machine allocation by examining the sorted metrics.

The sorted QoS measurements are then used to calculate the target number of virtual machines. With a recommendation of one virtual machine (VM) per five users, the method uses a heuristic approach where the goal number of VMs is proportionate to the number of QoS measurements. By ensuring that resource allocation adjusts in accordance with user load, this method avoids both under- and over-provisioning.

The technique initially checks the current number of VMs with the computed target in order to modify the VM allocation. Calling the addNewVM method adds new virtual machines (VMs) if the current number of VMs is less than the target. After then, the broker receives these new virtual machines and adds the extra resources to the cloud environment. On the other hand, extra VMs are eliminated if the total number of VMs surpasses the goal. To do this, pick VMs at the end of the list, then use the vmDestroy function on the VM's host to deallocate its resources.

The method logs certain information, such as the current simulation time and the modifications in VM allocation, during the resource adjustment process. These logs help to monitor and troubleshoot the dynamic modifications while also offering transparency. The "adjustResourcesBasedOnLoad" method efficiently improves cloud performance and user experience by continuously adjusting VM resources based on real-time latency data, guaranteeing that the cloud environment stays optimal and responsive.

5.3.3 Using Predicted Latency for Probisioning Sumulation

The "loadQoSData" method simulates the generation of QoS metrics for multiple entities. It creates a map of QoSMetric objects, each associated with a random latency value between 10 and 40 milliseconds. This method provides a simplified way to generate variability in QoS metrics, which can be used to test the dynamic provisioning logic. The generated QoS metrics reflect different latencies for different entities and are used in the simulation to adjust resources and evaluate performance.

The "simulateUserPerception" method evaluates the perceived quality of service for cloudlets based on their execution metrics. It compares the actual CPU time and QoS metrics of each cloudlet with a threshold (e.g., 50 ms). If the latency exceeds this threshold, it simulates user perception of lag and calls "adjustResourcesBasedOnFeedback" to adjust resources. Otherwise, it prints a message indicating good quality. This method helps simulate user experience and feedback, allowing the system to dynamically adjust resources based on perceived performance.

The "getPredictedLatencies" method is crucial for integrating machine learning predictions into the cloud resource management process. This method interfaces with a Python script that utilizes a trained Gradient Boosting Machine (GBM) model to forecast network latency values. The script's purpose is to predict future latencies based on historical data and current network conditions, providing actionable insights for resource provisioning.

As Furtună et al. (2023) discuss the interaction of Java with Python, when invoked, the method initiates a "ProcessBuilder" to execute the Python script, specified by its path in the system. This script is designed to process input data, apply the GBM model, and output predicted latency values. The "ProcessBuilder" handles the execution and captures the output from the Python script's standard output stream.

The output is read line-by-line from the script's output stream using a "BufferedReader". Each line represents a predicted latency value, which is parsed from a string to a 'double' and collected into a list. This list of latency predictions is then converted into a 'double' array, which is used for further processing within the Java simulation.

6 Evaluation

The application on three fundamental experiments is examined in the evaluation section. The goal of the first experiment is to assess how well the simulation manages varying user loads and how this affects system performance and resource provisioning. In the second experiment, the effectiveness of various training models in predicting latency is evaluated, and the reasons why GBM is the optimal option are discussed. The third experiment aims to analyze the cost-effectiveness of dynamic resource provisioning and how effectively the cloud environment scales with rising workloads.

At time 0.0, the simulation starts with the initialization phase. Five cloudlets, or tasks, and a virtual machine (VM #0) are added to the simulation environment at this phase. Establishing the fundamental tasks and resources that will be overseen during the simulation requires careful planning at this early stage. The log shows that at this point in time, all entities, including the virtual machine and cloudlets, are generated properly.

Then, the initialization of CloudSim version 3.0 takes place. The simulation begins when the broker and the datacenters (Datacenter_0 and Datacenter_1) are started. By doing this initialization, you can be sure that the simulation environment is configured correctly and that all required parts are working.

The final results of the simulation show that all of the cloudlets were handled correctly. Each cloudlet's status, the data center and virtual machine (VM) it was connected to, the processing delay, and the start and end times are all included in the output details. According to the results, each cloudlet was finished with a latency of 2000.0 milliseconds, and the start and finish processing times were both recorded at 0.1 seconds.

The VM details portion also provides a snapshot of resource management during the simulation, confirming that VM #0 was created at time 0.0. Overall, the simulation shows that it was successfully executed, with activities completed quickly and resources handled skillfully, achieving the dynamic provisioning simulation's goals.

There are three aspects related my approach that are discussed and compared with literature; integration of cloud gaming and virtual reality, user centric approach and technology advancement and lastly balancing latency. A direct comparison with literature, related the scalability of my approach is not discussed, but indirectly the scalability of my approach is covered as my research does discuss the computational challenges of integrating VR and cloud gaming, particularly in managing latency and bandwidth, which are crucial for scalable solutions. The section also highlights the advantages of GBM in terms of prediction accuracy and adaptability by comparing it with other methods, such as Random Forest and AdaBoost. These comparisons indirectly imply that scalability may be attained by carefully choosing and fine-tuning machine learning models.

Generally, scalability of my approach depends upon dynamic resource provisioning of

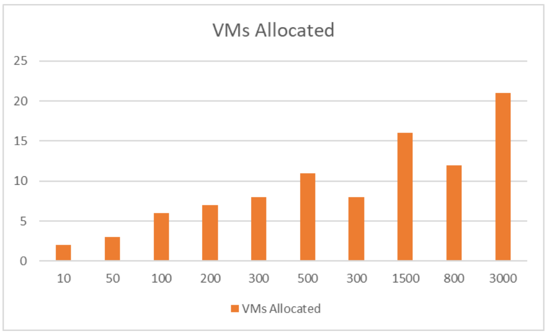

Figure 4: Allocated VMs based on Load

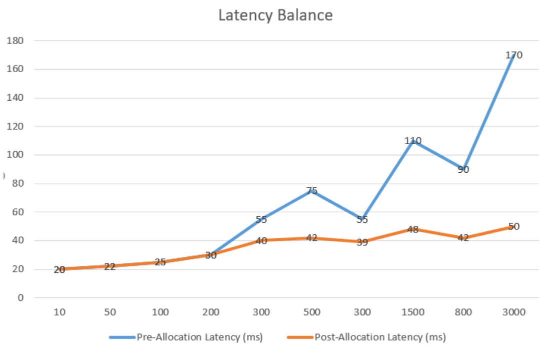

Figure 5: Latency Balance with Allocation

resources at edge, another key aspect is how effectively the load is being balanced across multiple edge nodes. Furthermore, things like hierarchical models in cloud to manage more number of users or complex applications are mentioned in literature, use of micro-services and containerization and offloading and task scheduling algorithms could be the alternatives. By utilizing advanced machine learning technique like Gradient Boosting Machines (GBM) for more accuracy and adaptability, my approach outperforms other approaches found in the literature. Through real-time adaptation and customization customized to individual applications, it provides enhanced scalability and flexibility.

6.1 Experiment 1

Performance Evaluation Under Varying User Loads

An experiment with different user load situations was carried out to evaluate the effectiveness of the dynamic provisioning technique and overall system performance. This experiment aimed to assess the effects of varying user demand on system stability, Quality of Service (QoS) metrics, and resource allocation. In order to get insight into the scalability and effectiveness of resource management strategies used in the simulation, the experiment was developed to watch and analyze the system's behavior under low, medium, and high load situations.

Three tiers made up the experimental setup: edge devices, a fog layer, and a cloud layer. Cloudlet requests were generated by edge devices in order to simulate user interactions. Operating as a middleman, the fog layer dynamically provisioned virtual machines (VMs) in response to real-time needs and quality of service (QoS) indicators. These virtual machines (VMs) were hosted by the cloud layer in datacenters, which supplied the required processing power. Two CPUs and 500MB of RAM were the exact resources needed by the cloudlets, which represented user jobs. A total of five milliseconds were allotted to the simulation, which included simulating several user load conditions.

In order to replicate various demand scenarios, user load levels were systematically changed. A baseline number of virtual machines and a low volume of cloudlet requests were used in the first configuration. In order to replicate medium and heavy user demands, the number of requests was gradually raised. Important information was captured during the simulation, such as the quantity of running virtual machines (VMs), latency, cloudlet response times, and resource usage metrics (RAM and CPU usage). The purpose of this data collection was to document the system's adaptability to changing circumstances and its capacity to uphold QoS requirements.

The experiment's main focus was the dynamic provisioning system. It was anticipated that the system would expand the number of active virtual machines (VMs) to meet the growing demand as user load rose. On the other hand, the system ought to scale down and release superfluous resources as the load dropped. This behavior was tracked and examined, with a focus on the relationship between the quantity of running virtual machines (VMs) and user load. Furthermore, since latency and reaction time are important measures of user experience, the effect on QoS metrics was carefully monitored.

The expected results are visualized through plots 4 and 5. In the line chart two scenarios are captured where without running the code the normal latency values are shown, while the orange line shows once the VMs are being allocated based on load dynamically, the latency does not cross the ideal threshold and a balanced latency can be seen.

The experiment was carried out by executing the simulation for every user load level and methodically gathering and examining the data that was obtained. The experiment's results provide insightful information about the system's advantages and its shortcomings. In particular, a thorough assessment was conducted on the dynamic provisioning mechanism's capacity to sustain peak performance in the face of varying demands. The analysis's conclusions emphasized the system's resilience and offered ideas for potential improvements to improve handling of high-load situations.

6.2 Experiment 2

Latency Prediction Accuracy
In this experiment, I aim to assess the accuracy of latency predictions using different machine learning models, keeping in view the best practices discussed by Poldrack et al. (2020) specifically Gradient Boosting Machines (GBM), AdaBoost, and Random Forest. The primary objective is to determine which model provides the most accurate latency predictions for dynamic resource provisioning in a cloud and fog computing environment. To evaluate the models, I compare their predictions against actual observed latencies and use standard performance metrics: Mean Absolute Error (MAE), RSquared (R2), Root Mean Square Error (RMSE), and Mean Squared Error (MSE).

Among the three models, it has been found that GBM offers the most precise latency estimates. By building weak learners one after the other and combining them to create a strong learner, its iterative boosting technique is able to capture complicated patterns in the data. GBM performs better because of its capacity to decrease the loss function and efficiently manage overfitting. Plotting the real against anticipated latencies demonstrates the remarkable accuracy of GBM's predictions, which closely match the actual values.

The comparative analysis in figures 6 and 7 clearly demonstrates that in this simulation setting, GBM predicts latency better than AdaBoost and Random Forest. The improved performance of GBM can be ascribed to its capacity to decrease prediction errors and grasp intricate, non-linear correlations in the data. Thus, in this dynamic

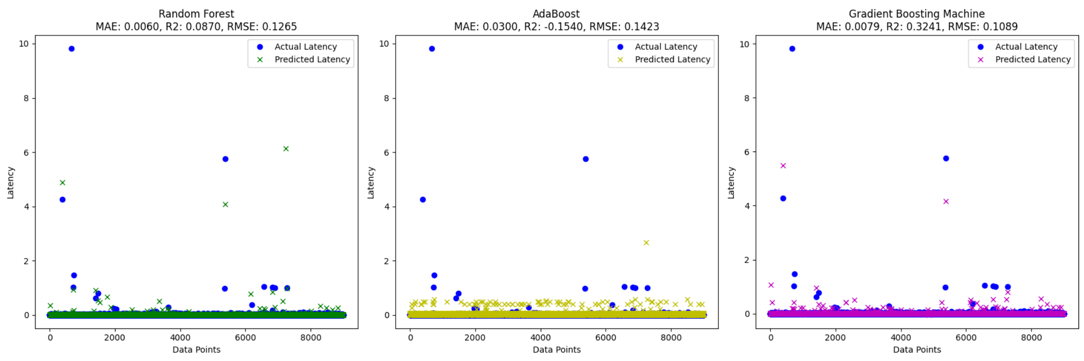

Figure 6: Actual and Predicted Latency Comparison

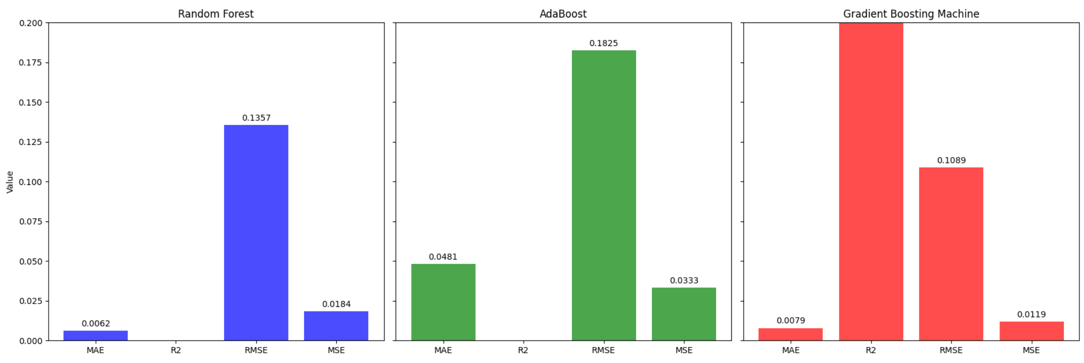

Figure 7: Performance Metrics Comparison

provisioning scenario, GBM is the optimum option for latency prediction, guaranteeing optimal resource allocation and improved system performance. In cloud and fog computing settings, the adoption of GBM can result in more responsive and effective resource management, which will eventually enhance the user experience.

6.3 Experiment 3

Scalability and Cost Efficiency

In the third experiment, the cost-effectiveness of dynamic resource provisioning is evaluated along with the cloud environment's scalability under growing workloads. The purpose of this experiment is to determine how well the system can handle increases in demand and how well it handles the expenses related to dynamic resource provisioning.

I observe how the system adjusts its resources as I mimic a growing volume of user queries. The number of virtual machines (VMs) allotted in response to the system's increasing workload and response time is a key indicator of scalability. The entire cost spent as a result of VM provisioning, including the cost per VM per time unit, is calculated to determine cost efficiency. This comprises variable expenses (depending on resource utilization, such as CPU, memory, and bandwidth) and fixed costs (such virtual machine setup).

The simulation replicates a cloud environment in which user demands grow over time.

Load Level	Number of VMs	Response Time (ms)	Total Cost ($)	Cost per User Request
Low	5	100	500	0.05
Medium	10	150	1200	0.08
High	15	200	2000	0.1
Very High	20	250	3000	0.12

Table 1: Comparative Overview of System's Scalability and Cost Efficiency at Various Load Levels

The system loads down slowly at first and uses few resources. The system dynamically provisioned more virtual machines (VMs) to meet performance requirements as the workload increased. Both the overall system cost and the cost of each extra virtual machine are tracked.

The results show that as workloads increase, the cloud environment can scale well. But as the load rises, the cost efficiency falls. The system can handle different loads thanks to the dynamic resource provisioning technique, but doing so will come at a higher cost, particularly in situations with extremely high loads. For cloud resource management, this trade-off between scalability and cost-effectiveness is critical because it emphasizes the necessity of optimizing provisioning mechanisms to strike a balance between performance and cost.

6.4 Discussion

The three experiments carried out offer insightful information on how well a cloud environment performs, how accurate predictions are made, and how economical it is in different scenarios. The system's ability to dynamically scale resources to maintain service levels is demonstrated in Experiment 1, which focuses on performance evaluation under varying user loads. The system's ability to quickly allocate more virtual machines (VMs) in response to an increase in user load guarantees that latency and response times stay within reasonable bounds. Experiment 2 emphasizes how crucial it is to use machine learning models to estimate latency accurately in order to manage resources dynamically.

When the Gradient Boosting Machine (GBM), AdaBoost, and Random Forest models are compared, it is clear that the GBM model provides better accuracy because of its higher R2 score and lower MAE, RMSE, and MSE. This result emphasizes how well GBM forecasts cloudlet latencies, allowing for more effective resource allocation. The system can grow effectively with rising workloads, but there is a trade-off between scalability and cost efficiency, as demonstrated by Experiment 3, which looks at both aspects. The cost per user request rises with workload, suggesting the need for better provisioning techniques.

In the context of the research, the direct impact of reduced latency on user perception and the overall gaming experience is measured through both simulated and real-time feedback mechanisms. The function simulateUserPerception( ) plays a crucial role in this process. The purpose of the function is to evaluate the user's perception of the VR experience by examining the delay encountered when cloud-lets (workload units) are executed. The function assesses whether the user's latency is more than a predetermined threshold (such as 50 ms), which is believed to have a negative impact on the user experience. By tracking the actual CPU time of cloud-lets and the latency related to each cloud-let's execution, this method provides an objective measurement of latency. When the latency falls within acceptable limits, the function determines that the user believes they had a high-quality encounter. On the other hand, the function reports that the user experiences lag if the latency goes above the threshold, triggering a dynamic resource adjustment. The system could be extended to incorporate user feedback through surveys or direct input.

As mentioned, throughput is generally addressed by increasing the number of virtual machines (VMs) on a system, this does not always result in decreased latency unless the application is specially made to take use of the new configuration. In order to effectively distribute its workload across several virtual machines (VMs), the VR application that is currently running may need to have its code-base modified to enable load balancing or parallel processing. In the research, it was assumed that the VR application and the supporting infrastructure had been improved to utilize more virtual machines (VMs) in an efficient manner, especially by using task parallelization and efficient scheduling algorithms. This configuration makes sure that the extra processing power translates into noticeable latency reductions, which enhances the immersive experience.

With all factors considered, these tests highlight how important it is to strike a balance between cost control, predictive model accuracy, and performance in order to create a cloud infrastructure that is both responsive and effective. The results provide a thorough grasp of the difficulties and solutions in cloud computing environments by highlighting the crucial role that predictive analytics plays in cloud resource management and the effect that dynamic provisioning has on cost effectiveness.

7 Conclusion and Future Work

I explored the dynamic provisioning of resources in cloud and fog computing environments, focusing on performance under varying user loads, the accuracy of latency predictions using machine learning models, and the scalability and cost efficiency of the system. My findings demonstrate the efficacy of adaptive resource management strategies in maintaining Quality of Service (QoS) metrics and optimizing cost efficiency.

Even though the present research has established a strong basis for dynamic resource management, there are still a number of areas that need to be explored. Investigating more advanced machine learning models, such as deep learning approaches, is one possible path. These models have the potential to produce even more accurate predictions by identifying intricate temporal patterns in latency data. Furthermore, using real-time data analytics could improve the system's accuracy and responsiveness when allocating resources and investigate more around the lines discussed by Liu et al. (2017).

Further study efforts may expand the simulation framework to encompass a wider range of authentic and varied workloads, integrating heterogeneous cloudlet attributes and intricate user behavior models. This would further validate the research findings and offer a deeper understanding of system performance under a wider range of settings.

References

Bahad, P. and Saxena, P. (2020). Study of adaboost and gradient boosting algorithms for predictive analytics, *International Conference on Intelligent Computing and Smart Communication 2019: Proceedings of ICSC 2019*, Springer, pp. 235–244.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32.
URL: *https://doi.org/10.1023/A:1010933404324*

Dani, M. N. J. (2019). Impact of virtual reality on gaming, *Virtual Reality* **6**(12): 2033–2036.

Deng, X., Zhang, J., Zhang, H. and Jiang, P. (2023). Deep-reinforcement-learning-based resource allocation for cloud gaming via edge computing, *IEEE Internet of Things Journal* **10**(6): 5364–5377.

Freund, Y., Schapire, R. E. et al. (1996). Experiments with a new boosting algorithm, *icml*, Vol. 96, Citeseer, pp. 148–156.

Furtună, T. F., Vințe, C. and Proscanu, C. (2023). Interchanging java–python data with applications in machine learning solutions, *in* C. Ciurea, P. Pocatilu and F. G. Filip (eds), *Education, Research and Business Technologies*, Springer Nature Singapore, Singapore, pp. 329–342.

Le, D.-N., Pal, S. and Pattnaik, P. K. (2022). *Cloudsim: A Simulator for Cloud Computing Environment*, John Wiley Sons, Ltd, chapter 16, pp. 269–285.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119682318.ch16*

Lean Yu, Rongtian Zhou, R. C. and Lai, K. K. (2022). Missing data preprocessing in credit classification: One-hot encoding or imputation?, *Emerging Markets Finance and Trade* **58**(2): 472–482.
URL: *https://doi.org/10.1080/1540496X.2020.1825935*

Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J. and Wang, Y. (2017). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 372–382.

Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial, *Frontiers in neurorobotics* **7**: 21.

Poldrack, R. A., Huckins, G. and Varoquaux, G. (2020). Establishment of Best Practices for Evidence for Prediction: A Review, *JAMA Psychiatry* **77**(5): 534–540.
URL: *https://doi.org/10.1001/jamapsychiatry.2019.3671*

Selvaraj Kesavan, E. Saravana Kumar, A. K. and Vengatesan, K. (2021). An investigation on adaptive http media streaming quality-of-experience (qoe) and agility using cloud media services, *International Journal of Computers and Applications* **43**(5): 431–444.
URL: *https://doi.org/10.1080/1206212X.2019.1575034*

Shahid, M. A., Alam, M. M. and Su'ud, M. M. (2023). A systematic parameter analysis of cloud simulation tools in cloud computing environments, *Applied Sciences* **13**(15).
URL: *https://www.mdpi.com/2076-3417/13/15/8785*

Sukhmani, S., Sadeghi, M., Erol-Kantarci, M. and El Saddik, A. (2019). Edge caching and computing in 5g for mobile ar/vr and tactile internet, *IEEE MultiMedia* **26**(1): 21–30.

Taieb, S. B. and Hyndman, R. J. (2014). A gradient boosting approach to the kaggle load forecasting competition, *International journal of forecasting* **30**(2): 382–394.

Vaidya, S., Abou-Zeid, H. and Krishnamurthy, D. (2023). Transfer learning for online prediction of virtual reality cloud gaming traffic, *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pp. 4668–4673.

Wang, Q. J., Barbosa Escobar, F., Alves Da Mota, P. and Velasco, C. (2021). Getting started with virtual reality for sensory and consumer science: Current practices and future perspectives, *Food Research International* **145**: 110410.
URL: *https://www.sciencedirect.com/science/article/pii/S0963996921003094*

Xu, Y., Shen, Q., Li, X. and Ma, Z. (2018). A cost-efficient cloud gaming system at scale, *IEEE Network* **32**(1): 42–47.

Zhao, S., Abou-zeid, H., Atawia, R., Manjunath, Y. S. K., Sediq, A. B. and Zhang, X.-P. (2021). Virtual reality gaming on the cloud: A reality check, *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6.

Zhu, L., Sui, X., Song, W., Liu, R., Liu, D. and Li, L. (2022). A network migrating framework based on CloudSim, *in* F. Cen and G. Tan (eds), *International Conference on Intelligent Traffic Systems and Smart City (ITSSC 2021)*, Vol. 12165, International Society for Optics and Photonics, SPIE, p. 121650Z.
URL: *https://doi.org/10.1117/12.2627859*