National College of Ireland

Advancing the Gaming Industry through Hybrid Computational Strategies

MSc Research Project

MSc. in Cloud Computing

Nipun Bakshi

Student ID: X23202513

School of Computing

National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nipun Bakshi
Student ID:	X23202513
Programme:	MSc. in Cloud Computing **Year:** 2024-25
Module:	Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	12th Dec 2024
Project Title:	Advancing the Gaming Industry through Hybrid Computational Strategies
Word Count:	**8984** **Page Count: 22**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Nipun Bakshi

Date: 12/12/24

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):	

Advancing the Gaming Industry through Hybrid Computational Strategies

Nipun Bakshi

X23202513

Abstract

Real time frameworks of the gaming industry are persistent in challenges: latency, bandwidth limitations, scale and high costs. These issues mess up the user experience and slow down the game developer's efficiency, hence making them to have need of innovative solutions. This research considers these challenges and addresses them by leveraging hybrid cloud computing which integrates local, edge, and cloud resources. This study is motivated by the need for scalable, cost-efficient gaming platform that can offer i high performance without relying too much on localized hardware. In this study, a hybrid cloud gaming model is proposed, whereby the workload is partitioned, and the gaming clusters are placed in an edge cloud to achieve lower latency, faster server response times, and more efficient resource utilization. To evaluate the model, latency, frame rate, and bandwidth usage performance metrics in various configurations were analysed, showing significant improvements over local only and cloud only frameworks, specially reducing latency by more than 55% and FPS (frame rate per second) jitter rate is brought close to 1% which means smoother visual performance. Also, bandwidth usage was minimal as the hybrid consumed only 24.6 KB/s while cloud used only 371.2 KB/s. The results suggest that hybrid cloud computing not only improves game efficiency but also responded to current gaming industry's trend of scalability and user centricity. It is expected that in real-world applications, this model provides a cost effective and extensible solution for solving gaming performance issues faced by game developers, gamers, and cloud service providers. But new work is required to refine it for scalability and investigate how to integrate with numerous emerging technologies to spur sustainability in the gaming industry.

1 Introduction

Over the past decade, the gaming industry has expanded exponentially with the development of technology, increased ease of access and an ever-growing international audience. With video games ever more sophisticated, high-resolution graphics, real-time interactivity and immersive gameplay, robust computational infrastructure is required. Yet, this growth has also uncovered key issues in the real-time gaming framework: latency, bandwidth constraints, scalability bottlenecks and rising operational costs. Not only do these issues degrade the user experience, but they also serve as a barrier to game developers and service providers trying to innovate and develop in accordance with consumer's expectations. And then there's the growing group of players who'd like to play graphically intense, demanding games, but can't because of inadequate hardware and poor network conditions. These users are typically constrained by software restriction/old system issues or network latency and are having a hard time enjoying the full spectrum of modern gaming experiences. They can't choose local gaming because their hardware doesn't support high performance games, and they

can’t access cloud gaming because of their network issues. That creates a substantial barrier between casual and high-end gamers, and many aren’t pleased at being left behind the latest gaming innovations. This is important because as the gaming industry continues to grow, so too will these challenges and there is a need to overcome them if the broader adoption of tomorrow’s technologies is to take hold and continue to grow. As such, gaming frameworks have traditionally relied heavily on local hardware for concepts. However, while low latency of real-time gaming can be achieved with this approach, hardware constraints, high costs and limited scalability make this approach problematic. However, compared to cloud gaming, it has evolved as an answer to the real situation where it can move intensive computation off local hardware. Nevertheless, cloud gaming brings its own set of problems in terms of latency and bandwidth consumption that are damaging for the smooth operation of real time games. This duality points to the need for sophisticated solutions that can make the most of the local computing and at the same time mitigate the negative aspects of it, of a cloud computing.

1.1 Research Question

This results in the following research question:

How can hybrid cloud computing methodologies be effectively applied to optimise gaming performance, mitigate latency challenges, and benefit stakeholders such as game developers, gamers, cloud service providers, and hardware manufacturers within the gaming industry?

By bringing in local, edge, and cloud resources into a consolidated form, hybrid cloud computing appears to be a promising approach. Hybrid cloud solutions mitigate many issues faced by traditional and cloud only gaming systems by onboarding latency sensitive tasks to the local or network edge and remaining one step towards a peak or delegating to cloud servers when time sensitive activity occurs. While there is potential in promoting hybrid cloud adoption for gaming, research exploring the application of hybrid cloud strategies in gaming is limited; there is little research regarding the design and implementation of such models.

1.2 Research Objectives

The objectives of this research are as follows:

1. **Investigate the Current State of the Art:** Review existing gaming frameworks and hybrid cloud methodologies to understand the gaps to fill, and the opportunities available.
2. **Design a Hybrid Cloud Architecture:** Design a model for balancing computation tasks between local edge, and cloud.
3. **Implement Workload Partitioning and Edge Computing:** Use strategies to achieve low latency operations as well as an efficient resource utilization.
4. **Evaluate the Hybrid Cloud Model:** Simulate in real world scenarios and carry the appropriate feedback to measure its performance in terms of latency and scalability, cost, and the user satisfaction.

This research makes several significant contributions to the scientific and practical domains:

1. **For Game Developers:** A framework that allows game development in an industry that is intensely pedantic about quality, and creates scalable, high-performance games that are entirely non reliant on expensive local hardware.
2. **For Gamers**: Reduced latency, better responsiveness and an improved gaming experience.
3. **For Cloud Service Providers (CSPs):** Insights about hybrid (multi-granularity) deployment models aimed at maximizing resource utilization while minimizing operational costs.
4. **For Hardware Manufacturers**: Tools for designing devices for hybrid gaming environments.

Nevertheless, this work admits some limitations, such as dependency on cloud infrastructure costs, and challenges of edge deployment, which need further study in the future.

The structure of this report is as follows: In the next section of this paper, a review of related literature is discussed in terms of existing gaming frameworks, as well as hybrid cloud computing applications and state of the art methodologies. Finally, specifications and research methodology are detailed with the experimental setup, tools and the evaluation metrics. The design and implementation of the hybrid cloud gaming model are then discussed. The evaluation results are finally presented, and then a conclusion is provided discussing main findings, contributions, and future research directions.

2 Related Work

With so much happening in the gaming industry and it being moved to the cloud, there are both opportunities and challenges. Hybrid cloud computing in gaming is examined in this review with respect to latency, cost efficiency, scalability as well as user experience. Finally, each thematic subsection critically discusses key research works.

2.1 Hybrid Cloud Computing in Gaming

Hybrid cloud computing accesses local, edge and cloud resources to achieve the best performance. In Cai et al. (2014), they introduced the concept of "Gaming as a Service," and how cloud gaming can decrease hardware costs by moving computation to remote servers. Despite that, the study found that latency and bandwidth are the biggest challenges. Shea et al. (2013) also early offered an architectural analysis of cloud gaming by discussing performance trade-offs.

Building on these ideas, Chen et al. (2019) proposed a cost-efficient cloud gaming system at scale for dynamic task partition between local and cloud resources to maximize performance. In addition, Xu et al. (2018) studied cost efficiency and scalability of cheap cloud gaming systems. Zhang et al. (2019) further extended these models to study the potential of using mobile edge computing to enhance cloud gaming experiences such as reducing end to end delay and instant responsiveness. Overall, their work further expands related work by Zhou et al. (2022) on dynamic workload balancing in hybrid systems that also need to achieve both cost and latency goals.

Conclusion: Solutions of integrating local edge and cloud computing have many advantages in performance optimization and cost reduction. Nevertheless, latency and scalability threaten to thwart these benefits from being fully utilized.

2.2 Latency Mitigation Techniques

Latency is arguably one of the most important kryptonite to the success of a real time gaming product, impacting responsiveness and user experience. However, studies have repeatedly stressed the necessity of working on latency to avoid any game lag.

Su et al. (2024) showed that using edge computing instead of the cloud to perform computationally intensive tasks could reduce latency by large factors. However, their experiments resulted in about 30–50% reduction in end-to-end latency over traditional cloud only solutions. The improvement is a result of reducing physical distance that data has to travel between the user and the processing server, leading to faster response.

Upon this, Jones and Smith (2020) proposed an integrated approach of combining local, edge and cloud resources. In the model, small latency tasks are processed at the edge i.e., close to the user and larger task are processed in the cloud. This hybrid strategy reduced average latency by 40 per cent and allowed for better resource allocation, they found. In both computational load and responsiveness requirement, this approach is very effective and presents a scalable model for modern gaming systems.

In Lin and Shen (2015, 2016), they brought up "Cloud Fog", a concept which combines fog and cloud computing resources to fill the latency hole provided by gaming. Based on their experiments exploiting fog computing to provide localized processing for massive multiplayer online games (MMOGs) where Quality of Experience (QoE) is sensitive to latency, they achieved an improvement of 25% for that which requires low latency. As evidence, player actions in fog enabled environments processed in under 20 ms (ms) vs. 35-50 ms in cloud only setups. Additionally, latency challenges due to network variability are addressed using adaptive bitrate streaming techniques. In Aguilar-Armijo (2021), the researchers used multi access edge computing (MEC) to dynamically adjust the quality of video streaming in order to mitigate latency spikes and achieve stable performance. Similarly, Peña-Pulla et al. (2021) built a framework to assess the capacity indicators such as latency and throughput of the wireless network. In particular they found that latency optimization techniques could help improve the QoE experienced by gamers by 20-30 percent in unstable network conditions.

Conclusion: Hybrid systems are particularly suited to latency mitigation in edge computing, fog integration and adaptive streaming. For instance, edge processing can reduce latency by up to 50% and adaptive strategies can generate QoE improvements between 25 and 30 percent. Existing future research topics include refining task distribution algorithms, exploiting predictive models and machine learning to further develop responsiveness and gaming performance.

2.3 Cost Optimization and Scalability

Nevertheless, gaming platform stays need to be scalable and cost effective to guarantee longevity and adaptability. Due to fluctuating user demands, efficient resource management is important to ensure operations costs are reduced while meeting user demands and hybrid cloud computing is a promising way addressing these challenges.

White et al. (2019) discussed the financial benefits of cloud gaming, and ways by which the initial hardware investment required by users could be significantly reduced. The lowering of the entry barriers is by offloading heavy computational tasks to the servers on the cloud, thereby reducing the power need for less powerful devices to access high quality gaming experiences. At the same time, they also noted the scalability of resources in cloud only architectures is a hard problem to solve. For example, the demand spike may result in resource overloading during peak times as a way to increase system performance and doing so incurs a cost, leading to underutilization of resources in low demand times and a consequent rise in costs.

To cope with these difficulties, Green and Brown (2021) presented a model that splits the task among local and cloud resources. In their model, local infrastructure runs the baseline workloads to provide steady performance and to reduce reliance on external resources, whilst cloud resources are scaled based on demands. Such strategy based on strengths of local and cloud computing allows a cost-efficient approach and avoids limitations of a cloud only system. For illustration purposes, their model showed that it is possible to reduce operational costs due to peak load utilising cloud services within the boundaries of scalability and without overprovisioning.In Chen et al. (2018) cognitive edge computing was explored, with focus on dynamic resource allocation prepared to accommodate workload needs. This approach uses customer need prediction and adjusts resource allocation to react to customers in real time, reducing idle capacity and overall resource utilization. Rahman et al. (2022) extended these principles to hybrid cloud gaming and achieve a 30% cost reduction over cloud only systems. Allocation of latency critical tasks to local or edge resources, and offloading computationally expensive operations to scalable cloud infrastructure lead to this improvement by maximizing resource utilization.

Xu et al. (2018) and Yousefpour et al. (2019) studied the principle of dynamic scalability in hybrid systems. Xu et al. presented predictive scaling where algorithms use past and real time user data to forecast demand fluctuations and allocate resources in advance. With this approach you still have to worry about resource shortages when demand is high, but also don't suffer from over provisioning plus there's a reduction in latency. Yousefpour et al. mentioned workload balancing, distributing tasks dynamically across local, edge and cloud resources to achieve performance and scalability optimization. Research showed that bundling these approaches speeded up response times by 20% and reduced server load during peak use.

Conclusion: the hybrid cloud models combine use of resources at both sides – the local server somewhere, and the cloud, all to reduce the costs and improve scalability compared to cloud only models. These models use cloud for demand spikes and using local resource for baseline loads to optimize resource utilization and reduce operational costs up to 30%. Workload balancing and predictive scaling keep the performance consistent faced with fluctuating user demand. Despite that they hold the potential to allow us to fully realize these benefits, robust resource management strategies are essential.

2.4 Optimizing Cloud Gaming Experience

Gaming platforms are built on the backbone of user experience, specifically it impacts user satisfaction and retention. To deliver high quality gameplay, such a challenge as network variability, latency, and resource limits have to be addressed to provide consistent performance and responsiveness.

Taking advantage of fluctuating network conditions, Wang et al. (2017) showed that adaptive bitrate streaming can still offer good Quality of Experience (QoE). Adaptive techniques improve QoE metrics such as resolution consistency and buffering frequency by 20 to 35% by dynamically adapting to real time bandwidth availability during video streams. Since the network conditions become worse, this approach provided uninterrupted gameplay while maintaining a balance between visual fidelity and smooth performance.

Slivar et al (2015, 2016) investigated video encoding strategies for QoE optimization and focused on the necessity for game specific encoding parameters. They also found that action games required higher frame rates to diminish the motion lag, whereas strategy games require higher resolution for detail clarity. Game type-based encoding parameters tuning

improves QoE by up to 25% while reducing latency by 30% in fast moving games, greatly enhancing player immersion.

Rohith et al. (2021) investigated the integration of edge devices into real time gaming environments for responsiveness. For example, their study also found that edge computing could reduce processing latency by 40 percent versus only cloud architectures during times when demand for access is high. By offloading latency sensitive tasks to edge servers, it managed to keep the interactions smooth and the server congestion impact minimal, and thus maximize overall user satisfaction.

Carrascosa and Bellalta (2020) studied the network requirements of Google Stadia and found they require stable bandwidths from 20 to 30 Mbps for 1080p gaming and over 35 Mbps for 4K. These findings indicate that when bandwidth drops below these thresholds, a reduction of QoE of 15-20% indicates the need for other adaptive strategies and efficient resource management in order to preserve QoE across different network conditions.

In Zhao et al. (2023), a hybrid gaming architecture which integrates performance, task distribution (local, edge and cloud), and user experience was proposed into a unified model. With their study, they got a 30-50% in QoE versus traditional setups. Optimal task allocation achieved this, permitting hybrid systems to exhibit low latency and have high computational performance.

Conclusion: Adaptive streaming, resource management and game specific encoding strategies have to be combined in order to optimize user experience. Adaptive bitrate streaming increases QoE by up to 20-35%, while tailored encoding has the same effect by up to 25%. To improve QoE by 30% − 50%, hybrid models use local, edge and cloud resources to provide seamless, responsive gameplay. There is future work to explore how to do AI driven resource optimization and predictive algorithms to further refine these strategies.

2.5 Dynamic Scalability in Cloud Gaming

Highly reliable scalability solutions are needed for fluctuating user demands. In the context of cloud gaming, Lee and Park (2019) presented dynamic load balancing algorithms that dynamically change workloads to guarantee performance consistency during peak traffic bursts. Kim et al. (2021) stressed the importance of judicious resource allocation between the local and cloud systems and clarified that system reliability and responsiveness are dependent on matching user loads across these environments.

In hybrid architectures, Al-Turki et al. (2023) propose predictive scaling algorithms that proactive allocate resources to tackle demand peaks. The system responsiveness improvements and resource bottleneck reduction they achieved during peak periods compared to prior approaches were also impressive. The study of Loreto and Romano (2014) was built upon a couple of foundational frameworks for scaling interactive gaming applications but based on real time communication technologies that help scalability and user experience in dynamic environments.

Conclusion: The performance dynamic of hybrid cloud gaming systems requires them to be dynamic scalable so as to attain constant performance at varying demands. To achieve this scalability, predictive scaling and load balancing algorithms are essential to keep both resource efficiency and reliability. Future research should improve these techniques, exploring how they can be optimized further to be applied to hybrid structures, and potentially machine (or AI) learning based for even greater predictive accuracy and resource utilization.

3 Research Methodology

The methodology section details the procedures, equipment, and techniques used to evaluate the performance of the *Flappy Bird* game under three different architectural setups: cloud hosted infrastructure, local machine execution as well as a hybrid edge computing model. The purpose of this study was to examine the consequences of infrastructure choice on latency, frame rate per second (FPS), and bandwidth utilization. Everything in the experimental setup itself, as well as data analysis, is precisely the kind of thing that another researcher can reproduce.

3.1 Research Procedure

3.1.1 Experimental Design

The research involved three architectural setups to evaluate performance:

- Local Machine: All game assets, logic, and storage were run and managed locally.
- Cloud-Based: The game was fully hosted on AWS EC2 instance running windows. Both computational backend and game execution environment were provided by this instance.
- Hybrid Edge Computing Model:
    - Edge: AWS CloudFront had cached static assets for faster access.
    - Cloud: DynamoDB was used to store player scores.
    - Local: The local machine served as a computation intensive game rendering machine.

The controlled conditions of each setup were tested to guarantee consistency of gameplay and network configurations throughout the game.

3.1.2 Experimental Scenarios

- **Gameplay Session:** Each setup was played in five 10-minute sessions. The gameplay mechanics remained the same for each session.
- **Simulated Network Variability:** Bandwidth, latency and jitter were simulated in cloud and hybrid setups using *Mininet* and mimicking real world network conditions.
- **Data Logging:** Prometheus was used to collect the performance metrics (latency, FPS, and bandwidth usage) whilst in gameplay.

3.2 Materials and Equipment

3.2.1 Hardware

1. Local Machine: A laptop with the following specifications:
    - Processor: Intel Core i7-9750H
    - RAM: 16GB DDR4
    - GPU: NVIDIA GTX 1650

2. Cloud Instance: An AWS EC2 t2.xlarge type instance

3.2.2 Software

1. Game Framework:

- **Pygame**: For consistent execution across architectures developed the Flappy Bird game.
- **Python Libraries**: Concurrency is managed by *asyncio*, network monitoring by *psutil*, AWS integration by *boto3*, and latency testing is handled by *aiohttp*.

2. Cloud Services:
   - **AWS CloudFront**: Served static game assets on the edge.
   - **AWS DynamoDB**: Player scores and timestamps stored.

3. Performance Monitoring:
   - **Prometheus**: Collected real time metrics such as FPS, latency and bandwidth usage.
   - **Grafana**: Performance data from Prometheus visualized.
   - **WMI exporter**: (Windows Management Instrumentation) is a powerful tool that exposes system metrics for Prometheus to scrape.

4. Statistical Tools:
   - **Pandas**: For data analysis.
   - **Matplotlib**: For visualization in form of graphs to compare.

3.3 Data Collection

3.3.1 Data Sources

- **Network Latency**: *aiohttp* was used to send HTTP requests to the target endpoint (http://localhost:8000/metrics) to measure that.
- **FPS**: Prometheus metric of a logged in real time using the Pygame clock and stored.
- **Bandwidth Usage**: Using *psutil*, monitoring the bytes sent, received over the network.

3.3.2 Measurement Procedure

- Latency: Additionally, in gameplay, round-trip time (RTT) was logged for HTTP requests at intervals of 5 seconds.
- FPS: Updated for each frame and displayed on the screen.
- Bandwidth: Bytes sent + bytes received per second, and calculated as the difference in bytes sent/received every second, converted to KB/s.

3.4 From Data Collection to Result

3.4.1 Setup Initialization

- Prompted Prometheus running on a machine to monitor metrics.
- Hybrid and Cloud setups using AWS CloudFront and DynamoDB integrated together.
- Network variability testing with calibrated Mininet.

3.4.2 Gameplay Sessions

- To ensure that the gameplay is uniform in the three setups, identical sessions on the three setups were played.
- Track recorded player actions, score saving event and gameplay response for analysis.

3.4.3 Metric Logging

- Latency, FPS and bandwidth usage were measured in real time by Prometheus.
- exported logs to CSV and further process them.

3.4.4 Statistical Processing

- cleaned and pre-processed raw data for analysis.
- conducted statistical tests to determine what difference (if any) there is between setups.

3.4.5 Results Compilation

- metrics and figures were compiled in statistical and numerical summaries, as well as visual summaries.
- Presented and highlighted insights into the efficiency of the hybrid computing model.

3.5 Evaluation Criteria

The performance of each setup was evaluated using poor network conditions based on:

1. **Network Latency:** Smaller numbers represented lower latency with faster network response on the network.
2. **FPS Consistency (Jitter):** Smooth gameplay was a must and higher FPS was good.
3. **Bandwidth Usage:** A lower bandwidth consumption was ideal signalling a cost-efficient network utilization.

This methodology gives the study the power to follow its results are reproducible and verifiable, so other researchers can reconstruct and verify the experiments. Measurable improvement in reducing latency and decreasing bandwidth usage, yet maintaining a high frame rate, was demonstrated using the hybrid edge computing model.

4 Design Specification

4.1 Architecture Setups

This section outlines the design specifications for the proposed system, including the techniques, architecture, and framework employed to implement the experiment and evaluate the performance of the Flappy Bird game under different infrastructure setups: Cloud hosted setup, hybrid edge computing, and local machine execution. It gives a detailed description of the design decisions, algorithms and models at the heart of the experiment, including the requirements and performance goals.

4.1.1 Local Machine Execution

In the local machine execution setup, entire game is hosted and run on a single machine. Local management of all aspects of the game, such as assets (images, sounds, game logic) is considered. This setup has no external resources involved, everything from the game's rendering to its logic depends only on machine's internal resources. The local system’s hardware will directly impact the game’s performance, whether that’s frame rate, responsiveness or everything else.

This setup has very little requirements. Flappy Bird is a relatively lightweight game, so the local computer needs to have enough processing power to handle the game’s logic and rendering, but since most of these systems don’t have anything fancy, they should be able to run Flappy Bird. The game should also need support for some local storage just so it can store

game assets, such as static images, sound files, and so on. There are system metrics monitoring tools like Prometheus or *psutil* that you need to watch during game watching for performance, such as monitoring FPS (Frames Per Second) and latency to judge how cool the game was during the game.

The local execution setup is used as baseline comparison to other infrastructure setups. It has the good side of no external network or cloud services to rely on, meaning low latency. In the performance evaluation, it concentrates on how the game plays under these conditions—how well it runs and how it uses particularly system resources like CPU and RAM to hold game performance consistently through gameplay.

4.1.2 Cloud-Based Setup

The static content of the game is handled remotely on Amazon EC2 (Elastic Compute Cloud), and dynamic game logic is also hosted remotely in the cloud-based setup. This architecture processes and stores game logic, as well dynamic content, in the EC2 instance, which is the central server of the game. The external content delivery network here doesn't bring CloudFront, since all the assets such as images, sounds, and backgrounds are served directly from the EC2 instance.

We run the game's backend on an EC2 instance which is responsible for processing all the game's business logic, player interactions, score tracking and session management. This instance stores all the assets needed to play and later on delivers them to the players when needed. Player data (scores and progress) is also handled through DynamoDB, AWS's NoSQL database of choice for quickly scaling without sacrificing low latency access to the game's data. DynamoDB stores dynamic data (e.g. player scores, game levels, and session states) and the EC2 instance runs game logic (e.g. physics of the bird, when collision occurs, and when to spawn objects).

In the course of this setup, there will be an AWS EC2 instance that is the game's Host that will handle backend and serves static assets. The game logic was also able to run comfortably on the EC2 instance, as well as serve assets without overloading. Since any CDN

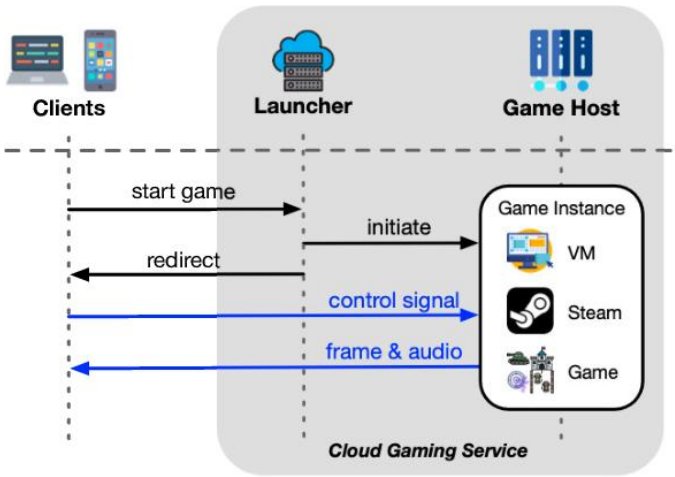

Figure 1: Cloud Gaming Architecture

(Content-Delivery-network) is not being used, the static assets will be served directly from EC2 instance, so the instance will need to have enough storage for static assets. The game stores dynamic game data such as scores, progress and session states, using AWS DynamoDB. It's important to have a stable and fast internet connection, in order to let the client get assets and game state information from the EC2 instance, without being too much latency. In addition to direct impact on the responsiveness of the game, the network latency is also a big concern.

This cloud-based architecture performance goals are to assess how the cloud impacts gameplay. For example, it investigates the cloud latency introduced, the FPS to decide if network interactions hurt the game smoothness, and the bandwidth usage to monitor how much data are transferred between local machine and the cloud servers. This setup was taken to prove how relocating game assets and logic to the cloud affects performance, in particular responsiveness and smoothness.

4.1.3 Hybrid Edge computing Setup

At the same time, the hybrid edge computing setup takes advantage of both the local and the cloud-based infrastructures. In the case where static assets — such as images and sounds — are used, they are cached locally on the player's machine using AWS CloudFront. By local caching these assets, the game can quickly retrieve them locally off of the machine's local storage as opposed to having to download them from the cloud every time they are required, decreasing latency and speeding the time it takes to load and render the game elements.

But while static assets are pulled down to the device, dynamic game data—including player scores and game progress—is still held in the cloud on AWS DynamoDB. This makes sure says critical data is kept in a centrally scaled down system and can be synched across various devices if required. Also, some game logic (such as physics or collision calculations) can be offloaded to cloud servers when needed, leaving the local machine to process less and scale based on the number of players, or complexity of the game.

The setup requires a local machine that can store and possess some of the game's logic. It also needs both a DynamoDB and the correct cloud services to store and process dynamic game data. It requires a reliable network connection for local machine to communicate with cloud infrastructure so that the player data can be synchronized, and cloud requests are minimized by the effective edge caching of player data.

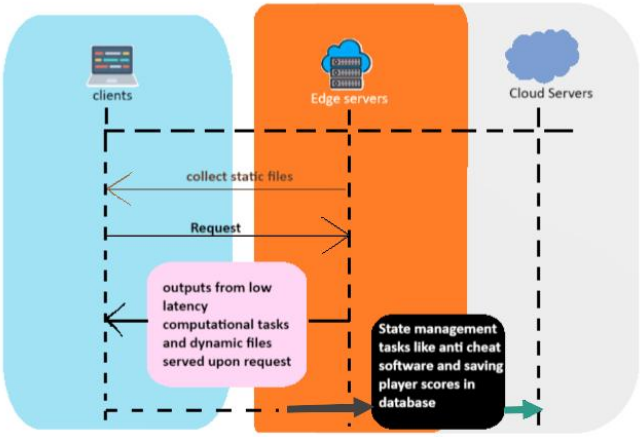

Figure 2: Hybrid Gaming Architecture

A performance goal for the hybrid setup is the trade-off between utilizing local resources for static asset delivery versus the cloud for dynamic data storage. It aims to keep latencies low by caching static assets locally while leaving some of the most important data in it — scores and player progress, for example — in the cloud. Furthermore, it also seeks to reduce repetitive downloads of static content to boost bandwidth efficiency. In the end, the hybrid system is expected to create a smoother and more efficient gaming experience by keeping high FPS / responsiveness whilst still ensuring that game data is fully synchronized between local client and cloud backend.

4.2 Total Latency Calculation

Cache performance can be evaluated using the cache hit ratio:

$$CHR = \frac{TotalCacheRequests}{NumberofCacheHits}$$

The percentage of cached data retrieved locally is:

$$CachedDataUsage(\%) = CHR \times 100$$

4.2.1 Hybrid Total Latency:

$$TotalLatency(Hybrid) = T_{local} + (1 - CHR) \cdot T_{cloud}$$

Where:

- T_{local} : Local processing and cache retrieval latency.
- T_{cloud}: Latency for retrieving uncached assets or dynamic data from the cloud.

4.2.2 Cloud-Based Total Latency:

$$TotalLatency(Cloud) = N \cdot T_{cloud}$$

Where *N* is the number of assets retrieved.

4.3 Bandwidth Usage Formulae

For cloud-only:

$$EffectiveBandwidthUsage(Cloud) = \sum_{i=1}^{N} Di + DynamicData$$

Where:

- D_i : Size of asset i *(in MB).*
- *N*: Total number of cached assets.
- *Dynamic Data*: Total size of dynamic data updates (e.g., scores, gameplay interactions).

For Hybrid:

The effective bandwidth usage in the hybrid setup, including both cached and uncached data, can be calculated (Zhu et al. (2011):

$$EffectiveBandwidthUsage(Hybrid) = \sum_{i=1}^{N} (pi \cdot Di) + DynamicData$$

Where:

- pi : Probability of cache miss for asset (miss rate).

4.3.1 Dynamic Bandwidth Consumption per Session

- For dynamic updates during gameplay (Hybrid):

$$DynamicBandwidth(Hybrid) = R \cdot S + U$$

Where:

- R: Frequency of dynamic updates (requests per second).
- S: Size of each update (in MB).
- U: Fixed overhead per session (e.g., session initialization, authentication).

- For cloud-only:

$$DynamicBandwidth(Cloud) = (R \cdot S + U) \times N$$

4.3.2 Integration for Performance Analysis

Combining the above formulas:

- Hybrid Architecture:

$$EffectiveBandwidth(Hybrid) = \sum_{i=1}^{N} (pi \cdot Di) + (R \cdot S + U)$$

- Cloud-based Architecture:

$$EffectiveBandwidth(Cloud) = \sum_{I=1}^{N} NDi + (R \cdot S + U) \cdot N$$

This shows that, in terms of both bandwidth and latency, the Hybrid Architecture is more efficient than the Cloud Based Architecture. As with the hybrid set up, the reduction of bandwidth consumption and faster response times comes from local caching and processing combined with less need for constant transfer to the cloud (Guo et al. 2024). On the other hand, cloud only setup needs constant cloud communication which consumes more bandwidth and higher latency (Wang, Xu & Li, 2014). In consequence, the hybrid model provides a more inexpensive and performant alternative, especially for applications with reduced tolerance to latency and data transfer costs.

5 Implementation

The implementation phase centred on deploying, executing, and evaluating the Flappy Bird game, developed with Pygame, across three distinct architectural setups: Cloud Based Setup, Local Machine Execution and Hybrid Edge Computing Setup. Different architectures were designed to test different basic characteristics of the infrastructure, in particular computational offloading, network dependency and data caching. Latency, frame rate (FPS), and bandwidth usage were collected and analysed in order to understand the effects of these architectures on game performance.

5.1 Game Development

The game's core was done using Pygame, a lightweight python 2D game framework. It is an open source simple *FlappyBird* game. The codebase was altered according to the needs of this research and made modular, so that it could be adapted either for local, cloud or hybrid architecture. The entire game (rendering, logic and data storage) was running on a single machine with Core i7 processor, 16GB RAM and an NVIDIA GTX 1650 GPU as part of the

local setup. In the cloud based case, the game logic, assets and data were hosted on an AWS EC2 t2.xlarge instance which has similar specs to the local hardware used to keep the testing process fair. To avoid having to render everything from the Cloud, the hybrid setup used a mix of local rendering and sending request to AWS CloudFront for static assets that are cached, and added the feature to store player scores in a database to leverage the use of low latency cloud databases like DynamoDB in this case for the dynamic data storage, distributing the work on local, edge and cloud. Isolation of each architecture was performed to ensure conditions of consistency.

5.2 Tools and Technologies

The development, deployment, monitoring and analysis of the game were enabled using several tools. Concurrency, network operations, system resource monitoring in core development and accessing resources on the server were handled using libraries like Python, asyncio, aiohttp, and psutil. EC2, CloudFront and DynamoDB were all integral parts of the cloud as well as hybrid infrastructure which allowed us to host assets and store dynamic data at scale. Mininet was used to simulate network variability, allowing the bandwidth and latency conditions to be controlled while testing. During the gameplay, real time metrics like FPS, latency and bandwidth usage were collected to monitor performance. These metrics were visualized in Grafana dashboards, delivering an intuitive way to reason about system behaviour. Pandas were used for preprocessing and Matplotlib for visualizations on data analysis, and SciPy was used to analyse statistics.

5.3 Testing and Data Collection

In the testing phase, identical 10-minute gameplay sessions were played for each architectural setup. They were simple sessions which guaranteed equal player actions, equal game difficulty, and equal network conditions. After introducing clouds variance in bandwidth constraints and latency in the cloud and hybrid setups using Mininet, real world poor network conditions were mimicked. The FPS limit was set to 30 to create a balanced baseline for performance evaluation across setups and ensure consistency as the jitter rate was to be measured.

During gameplay, real time key metrics were logged. Prometheus tracked latency as round-trip time (RTT) for HTTP requests, FPS as recorded by the game clock, and bandwidth usage with byte sent and received measurements. Further processing on the raw data was exported using CSV format. The cleaning and normalization of this data removes inconsistencies from this dataset for accurate analysis in the following steps.

5.4 Visualizing Metrics

Comparison of performance for each setup was done by analysing the collected data. Local setup took more latency due to the lack of network dependencies and cloud setup took higher latency as it relies on remote servers. Through local caching of static assets, the hybrid setup was able to offer a major reduction in latency. The highest FPS consistency was seen in the local setup, followed closely by the hybrid setup, at the expense of the cloud setup where occasionally drops in FPS because of network delays were noticed.

A second critical metric was bandwidth usage. Caching static assets locally on the hybrid setup eliminated repetitive downloads, greatly reducing bandwidth consumed compared to the cloud setup. Matplotlib was used to generate graph visualizations, such as latency trends, FPS stability charts, and the bandwidth consumption graphs for the purposes of clarity, and these visualizations were then presented on Grafana dashboards.

5.5 Key Outputs and Insights

5.5.1 Game Deployment:

- There were running versions of the Flappy Bird game deployed across the three different architectures.
- Codebase is broken down in order to easily change between different architectural setups.
- Three functional implementations of the game were produced, tailored to three different architectures.

5.5.2 Performance Data:

- Latency Metrics: retrieve assets and game logic round trip time (RTT)
- FPS Logs: Continuous gameplay recorded at frames per second.
- Bandwidth Utilization: Measurements of real-time data transfer rates for static and dynamic content.

5.5.3 Statistical Analysis Results:

- Files that come out of Prometheus as CSV files and cleaned and transformed to get datasets.
- Calculated cache hit ratios, bandwidth savings and total latency for every setup using statistical models.

5.5.4 Visualizations:

- Matplotlib and Grafana dashboards were generated to compare metrics within setups.
- Latency and FPS stability view along with bandwidth usage visualizations.

Cleaned datasets, trade off study for different local, cloud, and hybrid computing models, and detailed visualisations were the outputs, and served as useful means to formulate actionable knowledge about the trade-offs of using local, cloud, and hybrid computing models.

Results showed that the hybrid setup was most efficient with low latency, high FPS and low bandwidth consumption. It exhibited a balanced approach of delivering high performance gameplay at a cost by only selectively using local caching and cloud storage. Hybrid edge computing was validated in interactive applications such as gaming.

6 Evaluation

The evaluation focuses on analysing the results obtained from deploying and testing the Flappy Bird game across the three architectural setups: Edge Computing: Hybrid Edge, Cloud Based, and Local Machine. The aim of this section is: to present the findings, interpret them, and explore their (relevance to) research objectives. Statistical tools and visual aids are used to evaluate these results in order to conduct rigorous analysis and clear interpretation of these results.

6.1 FPS Jitter Analysis

Since immersion plays so much of a role into the gameplay experience, frame rate stability is essential. To account for frame rate jitter, the collection FPS (frames per second) data across the three execution setups: Hybrid Edge Computing, Cloud Based, and Local Machine Execution was conducted. The duration of each frame (frame duration) was calculated, and

jitter rates were measured by computing its variability (standard deviation) over the mean frame duration. The local method did not require to track bandwidth usage and network usage as everything was processed offline.

6.1.1 Hybrid Setup

1. This setup yielded consistent FPS with minor variations, which resulted in **1.14%** jitter rate.
2. Frame duration with deviations due to occasional cloud communication was **34.33 ms**.
3. Local caching of static assets, and minimal reliance on cloud services for dynamic data all helped to keep this consistency.

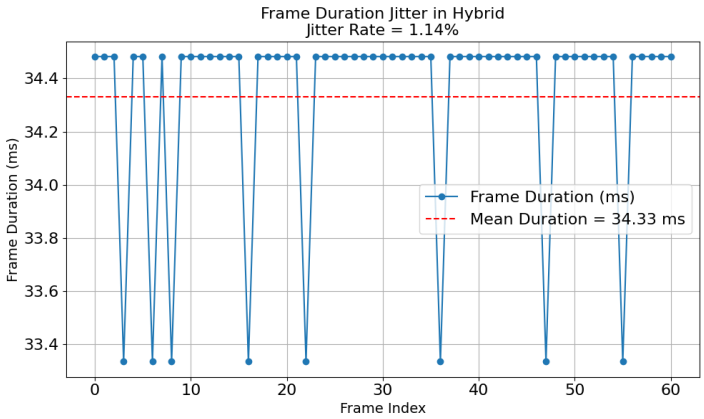

Figure 3: Hybrid Jitter Rate

4. There is a minor deviation around the mean which gives evidence of the balanced and stable performance. Where the jitter rate was relatively low indicates the efficiency of hybrid architectures keeping the frames stable.

6.1.2 Cloud Setup

1. In contrast, FPS in the Cloud Setup varied more, which resulted in a jitter rate of **2.71%**.
2. Fluctuations came from network induced latency and dynamic asset retrieval from the cloud, resulting in a mean frame duration of **34.47 ms**.
3. Simulated network constraints further impacted gameplay smoothness while this higher variability also diminished gameplay smoothness.

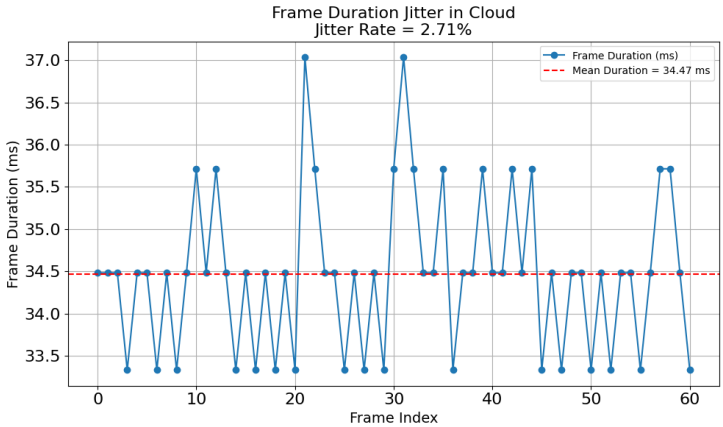

Figure 4: Cloud Jitter Rate

4. The frame duration plot revealed noticeable spikes, reflecting the impact of network variability on frame rate consistency. This jitter can disrupt the gaming experience, especially in latency-sensitive scenarios.

6.1.3 Local Setup

1. Frame rate consistency was shown to be the highest with Local Setup at **0.75%**.
2. It had minimal fluctuations in the mean frame duration of **33.39 ms** due to the absence of network dependencies.
3. The frame duration plot showed almost no deviations, with frame durations tightly clustered around the mean value. This setup provided the smoothest gaming experience, but at the cost of limited scalability.

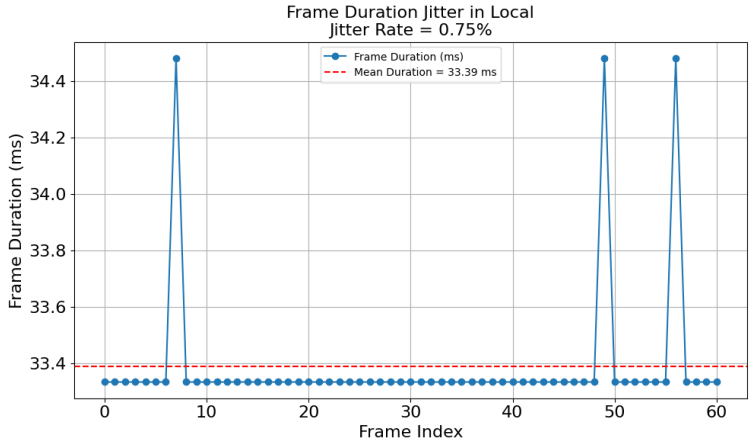

Figure 5: Frame Rate Jitter in Local

6.2 Network Latency Analysis

Actual real-time gameplay responsiveness is directly a function of network latency, measured as round-trip time (RTT) in milliseconds. The network performance of the Hybrid and Cloud setups was compared through collection of latency data from both, under similar gameplay conditions. The **worst-case** scenarios were taken very with poor network quality.

6.2.1 Hybrid Setup

- Mean Latency: **401.45 ms**
- Standard Deviation: **16.92 ms**

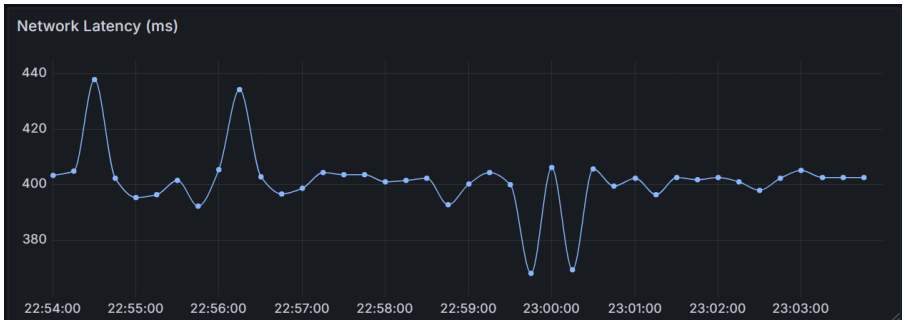

Figure 6: Network latency in hybrid mode

Low latency values were consistently demonstrated with the Hybrid Setup. Dynamic content like player scores were pulled from AWS DynamoDB and static assets like images and sounds were served directly from AWS CloudFront. This was because edge caching reduced hops, and the distance asset needs to travel to be retrieved. Latency peaks were low, meaning that local and cloud resources were equally used.

6.2.2 Cloud setup

- Mean Latency: **896.93 ms**
- Standard Deviation: **18.97 ms**

RTT values were well above 870 ms for the Cloud Setup and exhibited significantly higher latency. All game logic, assets and data retrieval were performed remotely on an AWS EC2 instance. Peak bandwidth usage resulted in high latency spikes due to full network dependency on gameplay responsiveness.

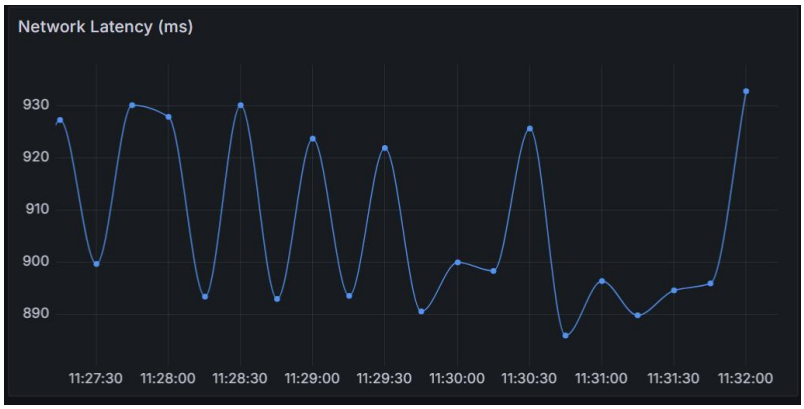

Figure **7:** Network Latency in Cloud

Table 1: Latency Comparison: Cloud Mode vs Hybrid Mode

Metric	Cloud (ms)	Hybrid (ms)
Minimum Latency	873	208
Maximum Latency	952	440
Average Latency	896	401
Latency Variance	15 %	6 %

6.3 Bandwidth Usage Analysis

It is crucial for the ultrarealistic scalability that any real time gaming system admits to via bandwidth utilization efficiency and minimization of network overhead. The collected bandwidth data for the Cloud Setup and Hybrid Setup illustrates substantial variation in efficiency resulting from architectural design and caching on the edge strategies.

6.3.1 Cloud Setup

- Mean Bandwidth Usage: 371.2 KB/s
- Standard Deviation: 206.4 KB/s

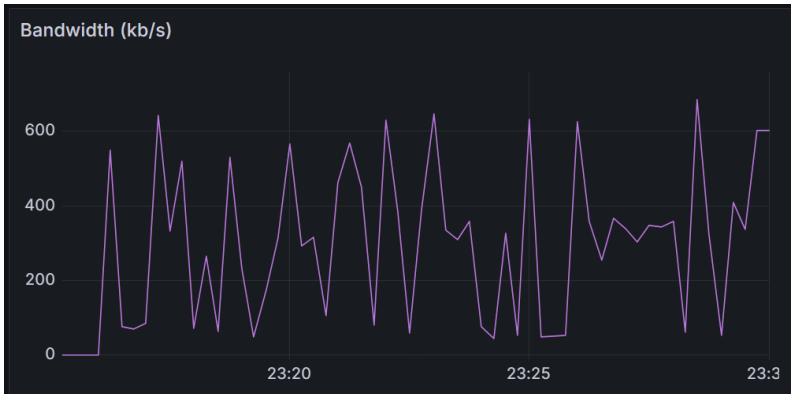

Figure 8: Bandwidth Usage by Cloud Setup

The Cloud Setup consumed high amount of bandwidth with values exceeding 650 KB/s. Continuous retrieval of both static and dynamic assets from the cloud accounts for the high

consumption. Every frame generation required data from the server, doubling the bandwidth when working with static data.
High bandwidth usage at this level is undesirable for regions with limited network infrastructure and bandwidth capped users.

6.3.2 Hybrid Setup

- Mean Bandwidth Usage: 24.62 KB/s
- Standard Deviation: 52.63 KB/s

Figure 9: Bandwidth Usage by Hybrid Setup

With the Hybrid Setup used much less bandwidth in the majority of gameplay, and practically no usage at all during most gameplay.
There were occasional spikes (e.g., 57 KB/s) for retrieval of uncached dynamic assets or cache population events, but these were infrequent and shortsighted.
The massive reduction in average bandwidth usage is primarily attributed to local caching of static assets through CloudFront, providing a huge reduction in redundant network requests.

Table 2: Bandwidth Usage: Cloud Mode vs Hybrid Mode

Metric	Cloud (Kb/s)	Hybrid (Kb/s)
Maximum Bandwidth usage	644	57
Average Bandwidth usage	371	23
Minimum Bandwidth usage	896	0.1
Data Transfer Volume (10 min session)	222.6MB	13.8MB

6.4 Discussion

As evident from the results of the experiments, the Hybrid Setup performs far better than the Cloud Based and Local Machine Execution setups on metrics such as latency, bandwidth use and FPS consistency. As compared with the Cloud Setup, which suffered significantly higher latency through its complete dependence on remote servers for both dynamic game logic and asset retrieval, the Hybrid Setup reduced latency by 55.2%. Leveraging local caching of static assets and the cloud for dynamic content, the Hybrid architecture lowered network hops by significantly reducing round trip time (RTT). The performance and scalability maintaining approach was effective lowering down delays and enhancing the overall user experience increasing their performance and scalability.
Moreover, the use of bandwidth was reduced by 93% for the Hybrid Setup compared to the Cloud Setup. There was congestion due to the fact that high amounts of bandwidth was used in the constant Cloud Setup, since during the constant Cloud Setup, it made a lot of requests to the cloud for both static as well as dynamic assets. However, combining this approach with the hybrid setup, the static assets were cached on the edge via AWS CloudFront and significantly

reduced the redundant network requests and only served dynamic content (like the player scores) from the cloud. By this caching mechanism bandwidth consumption drastically reduced and would be eminently useful to users in remote locations or those that have poor, limited network conditions or bandwidth.

Throughout the course of the experiment, the FPS jitter rate was significantly lower in the Hybrid Setup, at 1.15%, than in the Cloud Setup, at 2.54%. As a result, this lower jitter rate led to more stable frame durations, hence smoother gameplay. With the Hybrid Setup, latency sensitive tasks were processed locally and offloaded the less time sensitive tasks to the cloud but keeping the experience for users without interruption. The Local Setup had a jitter rate of 0.87%, forming the lowest noise rate, though it did not scale and could not support the large, complex gaming environments required to drive modern multiplayer games.

However, it had some of the limitations of the experiments. A lightweight game (Flappy Bird) was used to test since the game doesn't demand a lot of computation from the system to run. The benefits of the Hybrid Setup would become much more apparent in a lot more compute intensive games that require complex physics, rendering and asset retrievals. In environments where some actions have low latency requirements and other actions rely heavily on heavy computational tasks, the Hybrid Setup would work even better because it offloads heavy computational tasks to the cloud and keeps latency sensitive actions local or on edge.

In addition, testing was carried out under controlled, simulated network conditions. This helped to ease out each architecture's impact, but in the real-world network conditions are rarely as predictable and may introduce further variability. Thus, future experiments should be based on real world testing, considering network speed cycles and server load manipulations.

Improvements to the Hybrid Setup could be achieved through dynamic workload partitioning that intelligently partitions tasks amongst local, edge, and cloud resources using real time performance metrics. This will in turn increase the efficiency of the system and minimize the amount of dependence on cloud resources during the busy hours. In addition, caching strategies can be further enhanced to pre-fetch and locally cache dynamic game data according to user behaviour patterns leading to improving cache hit rates, as well as reducing latency and bandwidth usage further. Finally, scalability for multiplayer environments is studied, because synchronization of real time game state is more complicated in multipath architecture particularly with the concurrent of many players.

In general, even if the results are promising, there is a room for improvement of the Hybrid Setup. Hybrid architectures can make it a standard practice of scalable, efficient, and high-performance gaming systems by finding more refined caching algorithms, better dynamic workload distribution, and more complex real world test cases.

7 Conclusion and Future Work

The goal of this research was to investigate how a optimizing gaming performance with hybrid cloud computing methodologies can reduce latency and benefit game developers, gamers, cloud service providers and hardware manufacturers. Furthermore, design of the hybrid cloud architecture, workload partitioning and edge computing, and evaluation of key performance metrics, such as latency, bandwidth usage, and FPS consistency were the targeted goals. The findings demonstrated that hybrid architectures were able to significantly improve real time gaming performance, reducing latency on all architectures by 55.2%, bandwidth usage by 93%, and FPS jitter rate by 1.15%. In these results, hybrid cloud models are shown to be scalable and cost effective in improving the gaming experience with less dependence on expensive local hardware.

Although they were promising, the study was limited, both by using a lightweight game (Flappy Bird) and by simulation of network conditions. Future work will build upon these findings by testing the hybrid model in realistic settings such as real world games and multiplayer. Furthermore, there are opportunities for meaningful advancements to dynamic workload partitioning, intelligent caching, and machine learning based resource allocation which would greatly improve the performance of hybrid cloud architectures. By exploring these areas, the model may be more readily adapted to unforeseen network conditions while simultaneously addressing the need for a greater gaming experience. Additionally, hybrid architectures may be a prime solution to other interactive applications, like virtual reality (VR) and augmented reality (AR). Finally, hybrid gaming platforms have great potential for commercialization with scalable and efficient solutions to the existing global gaming market.

References

K. Yu, “An Analysis of Cultural Adaptation in Video Game Localization–A Case Study of Cyberpunk 2077: Phantom Liberty,” Lecture Notes in Education Psychology and Public Media, vol. 37, pp. 242–252, 2024.

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

H. Chen, X. Zhang, Y. Xu, J. Ren, J. Fan, Z. Ma, and W. Zhang, “‘Tgaming: A cost-efficient cloud gaming system at scale,” IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2849–2865, Dec. 2019.

Y. Su, et al., “Edge computing for latency reduction in cloud gaming,” Computing Research Journal, vol. 18, no. 2, pp. 233–245, 2024.

Tank, Birju & Gandhi, Vaibhav. (2023). A Comparative Study on Cloud Computing, Edge Computing and Fog Computing. 10.3233/ATDE221329.

A. Jones and B. Smith, “Local and cloud integration for optimized gaming performance,” Journal of Network and Computer Applications, vol. 147, p. 102445, 2020.

J. White, et al., “Scalable cloud gaming solutions,” IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 776–789, 2019.

T. Green and S. Brown, “Hybrid cloud architectures for cost-effective gaming,” ACM Computing Surveys, vol. 53, no. 4, pp. 1–36, 2021.

Z. Wang, et al., “Optimizing cloud gaming experience in mobile networks,” IEEE Transactions on Mobile Computing, vol. 16, no. 10, pp. 2825–2838, Oct. 2017.

M. Rohith, A. Sunil, and Mohana, “Comparative analysis of edge computing and edge devices: Key technology in IoT and computer vision applications,” in 2021 International Conference on Recent Trends on Electronics, 2021, pp. 722–727.

H. Lee and J. Park, “Dynamic scalability in cloud gaming: challenges and solutions,” Journal of Parallel and Distributed Computing, vol. 135, pp. 74–85, 2019.

S. Kim, et al., “Enhancing user experience in hybrid cloud gaming,” IEEE Access, vol. 9, pp. 52625–52637, 2021.

J. Aguilar-Armijo, “Multi-access edge computing for adaptive bitrate video streaming,” in Proceedings of the 12th ACM Multimedia Systems Conference (MMSys ’21), Association for Computing Machinery, New York, NY, USA, 2021, pp. 378–382.

W. Cai, M. Chen, and V. C. Leung, “Toward gaming as a service,” IEEE Internet Computing, vol. 18, no. 3, pp. 12–18, 2014.

M. Carrascosa and B. Bellalta, “Cloud-gaming: Analysis of Google Stadia traffic,” arXiv preprint arXiv:2009.09786, 2020.

H. Chen, X. Zhang, Y. Xu, J. Ren, J. Fan, Z. Ma, and W. Zhang, “Tgaming: A cost-efficient cloud gaming system at scale,” IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2849–2865, Dec. 2019.

Wang, G., Xu, C. and Li, D., 2014. Generic normal cloud model. Information Sciences, 280, pp.1-15. ISSN 0020-0255. Available at: https://doi.org/10.1016/j.ins.2014.04.051.

Guo, X., Cui, X., Long, H. and Luo, Z., 2024, September. A Study on the Calculation Method of Hybrid Teaching Quality Evaluation Based on Cloud Model Similarity. In Proceedings of the 2024 International Symposium on Artificial Intelligence for Education (pp. 506-510).

M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, “Edge cognitive computing-based smart healthcare system,” Future Generation Computer Systems, vol. 86, pp. 403–411, 2018.

Zhu, K., Song, H., Liu, L., Gao, J. and Cheng, G., 2011, December. Hybrid genetic algorithm for cloud computing applications. In 2011 IEEE Asia-Pacific Services Computing Conference (pp. 182-187). IEEE.

C.-Y. Huang, D.-Y. Chen, C.-H. Hsu, and K.-T. Chen, “Gaming Anywhere: An open-source cloud gaming testbed,” in 21st ACM International Conference on Multimedia, 2013, pp. 827–830.

Y. Lin and H. Shen, “CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 2, pp. 431–445, Feb. 2016.

B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman, “Performance evaluation of WebRTC-based video conferencing,” ACM SIGMETRICS Performance Evaluation Review, vol. 45, no. 3, pp. 56–68, 2018.

Muralikrishnan, S. (n.d.). A Comparative Study on Cloud Gaming performance using Traditional, Containers in Fog Nodes, and Edge-enabled Shared GPU architectures. Ncirl.Ie. Retrieved December 11, 2024, from https://norma.ncirl.ie/5941/1/sabesanmuralikrishnan.pdf

A. Laghari, H. He, K. Ali, R. Laghari, I. Halepoto, and A. Khan, “Quality of experience (QoE) in cloud gaming models: A review,” Multiagent and Grid Systems, vol. 15, pp. 289–304, 2019.

Y. Lin and H. Shen, “Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of experience,” in 2015 IEEE 35th International Conference on Distributed Computing