Optimizing Latency and Cloud Dependency in Edge-Cloud Distributed Systems Using iFogSim

Aryan Murugesh
Department of ECPE
Iowa state university
(aryanm54@iastate.edu)

Abstract-*Depending just on the cloud becomes ineffective when demand for real-time processing rises, particularly in applications like VR gaming, driverless cars, and healthcare monitoring. Users suffer increased delay the farther data moves, which causes major latency problems. Increased network congestion, energy usage, and operating expenses follow from cloud-only solutions as well. This work uses fog computing using iFogSim—a simulation toolkit—to investigate more efficient solutions by assessing how shifting compute near the data source can lower latency and general cloud consumption.*

By means of simulations of an EEG-based VR game, we evaluate a conventional cloud architecture versus a fog-based implementation. The application demonstrates amazing results when modules like connection and concentration_calculator are handled at fog devices instead of the cloud. Latency, energy use, and cloud expenses all dropped sharply. This work presents those results together with supporting detailed graphs, logs, and screenshots. We also present a fresh dynamic latency-aware module placement technique to replicate smart resource allocation in edge-cloud systems.

I.INTRODUCTION

Many modern systems, especially those requiring real-time interaction, face performance bottlenecks when relying solely on centralized cloud infrastructure. While the cloud offers high computational power, its physical distance from end users causes significant delays. Applications like virtual reality games, autonomous driving, or remote patient monitoring demand ultra-low latency and rapid response times.Fog computing addresses this challenge by introducing a layer between end devices and the cloud. It brings computation and storage closer to users, utilizing intermediate devices such as gateways, routers, and microservers as fog nodes. These nodes process data locally, reducing round-trip times, easing network congestion, and cutting costs.

This paper presents a hands-on simulation using iFogSim to explore the potential of fog computing in optimizing latency and cloud dependency. Using an EEG-based multiplayer VR game scenario, we analyze how different application module placements impact system performance. The results demonstrate the tangible benefits of edge computation, including faster processing, lower energy usage, and reduced bandwidth consumption.

II.RELATED WORK

As substitutes for centralised cloud models, fog and edge computing have been investigated several times. First presenting the idea of fog computing, Bonomi et al. [1] underlined how it helps to increase cloud capabilities near end users. Later studies, notably Gupta et al. [2], offered iFogSim as a strong tool for simulating resource management in fog environments.

Other publications have suggested different module locations with an eye towards cost, energy, or performance. Mahmud et al. [3] categorised fog computing techniques as either static or dynamic, demonstrating that although basic, stationary placement lacked flexibility. Though they lacked latency-based measurements, Aazam and Huh [4] proposed dynamic models based on resource availability.

Our study expands on these foundations by using a latency-driven placement method, therefore mimicking a more realistic, flexible edge-cloud system. Few

simulations, to the best of our knowledge, in iFogSim mix loop latency data with automated module deployment decisions.

III. SYSTEM DESIGN

The system is built as a three-layer edge-fog-cloud hierarchy. End devices as cellphones and EEG sensors make up the base layer. These devices send data to gateway fog nodes, which stand in for modestly computationally capable local servers. The highest tier stands for centralised cloud data centres.

We replicate a VR game in real time multiplayer. Every player has an EEG headgear that feeds brainwave data to a smartphone, which forwards the data to fog nodes for processing. Three main modules define the game:

1. **Client Module:** Collects EEG data, sends to processor, receives feedback.
2. **Concentration Calculator:** Analyzes EEG signal for concentration levels.
3. **Connector Module:** Aggregates and synchronizes game state between players.

One can arrange each module at several layers. All modules under the cloud-only model are located there. The fog-static paradigm positions the Client module at the cloud, Concentration Calculator at gateways, and Connector at smartphones. Measuring uplink latency from fog devices, the dynamic placement model uses the Concentration Calculator and Connector to reach the lowest latency nodes.

Modifying the ModulePlacementEdgewards.java in iFogSim allowed latency-aware placement to sort available fog nodes depending on their simulated uplink latency and assign modules accordingly.

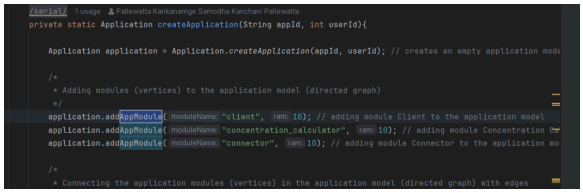

Fig-1:Module to device mapping view in iFogSim

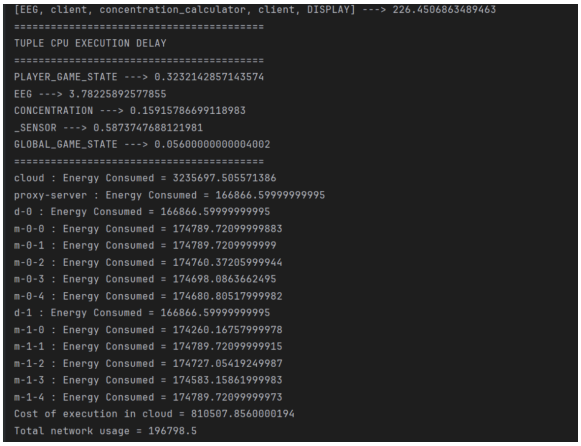

Fig-2 : Output after running it on cloud without any edge computing.

IV. METHODOLOGY

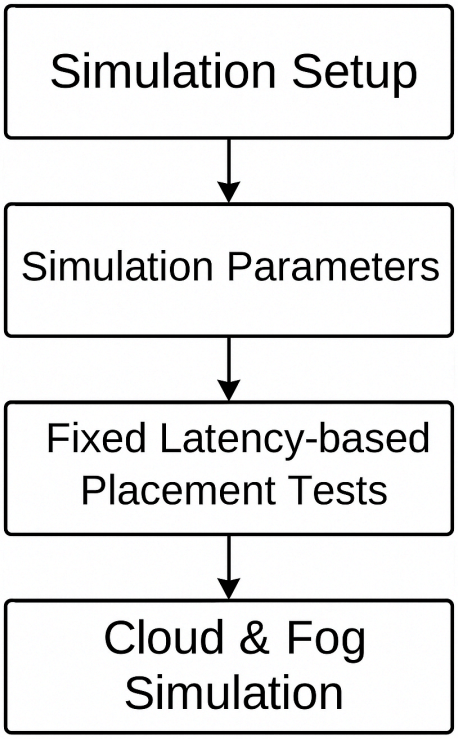

Fig-3:Methodology of my project

We modelled and simulated fog and edge computing settings using iFogSim, a strong toolkit meant for this purpose. This system's main goal was to reproduce a reasonable real-time distributed system stressing the effects of module location on general latency, energy consumption, and network use. We developed a layered and ordered topology reflecting a simplified real-world situation to provide a relevant and equivalent analysis.

Mobile devices, fog nodes, and cloud data centres formed a three-tier architecture guiding the simulation environment. We modelled two departments, depicting industrial units or hospital wings as physical surroundings. Every department consists of five mobile users with an EEG sensor sending brainwave information. These mobile nodes represent lightweight edge devices—such as tablets or smartphones—that act as the initial layer of data collecting and pre-processing.

One proxy server serves as the link between fog gateways and the centralised cloud infrastructure in the middle layer. In large-scale networks, this proxy node models reasonable latency and bandwidth constraints. It essentially controls packet routing by forwarding data both upward to the cloud and below to the fog.

Every stage in the hierarchy received uplink and downlink latencies to properly represent communication delays. For instance, gateways to proxy at 4 milliseconds, mobile nodes and fog gateways set their latency to 2 milliseconds, and proxy to cloud at 100 milliseconds. Seeking to replicate real-world transmission properties, these delay figures are determined from common industrial installations and published literature.

Furthermore modelled using deterministic distribution patterns was data transmission behaviour from EEG sensors. Based on clinical investigations and signal capture intervals for brain-computer interfaces, this reflects a constant data output rate. We reduced unpredictability by employing deterministic inter-arrival durations, therefore concentrating on controlled performance variance across installations.

Every node—mobile, fog, proxy, cloud—was set up using hardware guidelines including MIPS (million instructions per second), RAM, bandwidth, and energy models. For instance, gateways had 2800 MIPS and greater RAM, mobile nodes were assigned 1000 MIPS and 270 units of bandwidth, while cloud nodes showed 44800 MIps. These parameters enable accurate simulation of processing capability.

Our experiments revolved mostly on module positioning techniques. We tested and used three placement models:

- Every application module—including the Client, Concentration Calculator, and Connector—was housed on the cloud node. With maximum predicted delay and cloud energy consumption, this formed the baseline model.
- .Under this arrangement, the Client module was housed on mobile devices, the Concentration Calculator on gateway fog nodes, and the Connector stayed on the cloud. With fixed, manually determined mappings, this model shows present edge computing techniques.
- This new method classifies available candidates and computes the uplink latency from every fog device in real time. Dynamic assignment of modules including the Concentration Calculator and Connector to the fog nodes with lowest uplink latency was done Customising ModulePlacementEdgewards.java in iFogSim helped one to apply the logic. This approach brings intelligent, adaptive behaviour into place of placement choices.

Using IntelligJ IDE in a Windows 10 environment, all simulations were run with output logs, module locations, and latency metrics noted for later review. For inclusion in this report, screenshots were obtained at every major point—initial configuration, module placement output, and execution outcome summary.

Fig-4: Code after moving it from cloud to edge computing

V. RESULTS

Our simulations yield a thorough comparison spanning several performance criteria. Energy use, latency, tuple execution delays, and total network cost guided each configuration—cloud-only, static fog, and dynamic

placement. Plot graphs and bar charts help to visually clear the results.

1. Energy Usage: Under the cloud-only arrangement, the centralised cloud carried all of the computational load, leading to an energy consumption level more than 3.2 million units. By contrast, the total energy usage reduced to 2.66 million units when modules were offloaded to fog nodes in the static fog placement model, therefore demonstrating a notable increase in efficiency. The decrease was mostly due to fog nodes handling a significant amount of the application load; they are geographically closer to the source and use less transmission power. Using fog infrastructure, energy consumption dropped down the hierarchy and more effectively balanced power draw.

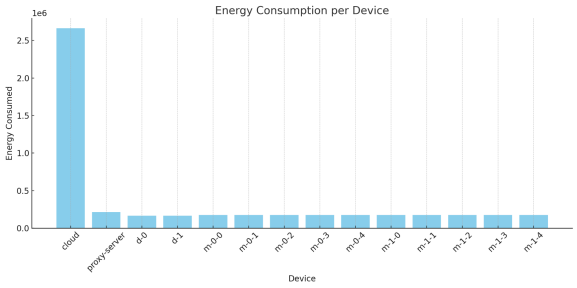

Fig-5::Energy consumption bar chart across device layers.

2. Latency: A major performance metric in our simulation was latency. Comprising EEG detection, processing, and actuator feedback, the application cycle completed in the cloud-only environment over 1125 milliseconds. This delay fell significantly to 226 milliseconds when the modules were moved to fog devices using the static placement policy, proving the advantage of edge-based deployment in lowest round-trip delays. Time-sensitive applications such health monitoring and autonomous control systems depend on low latency since dependability depends on instantaneous response.

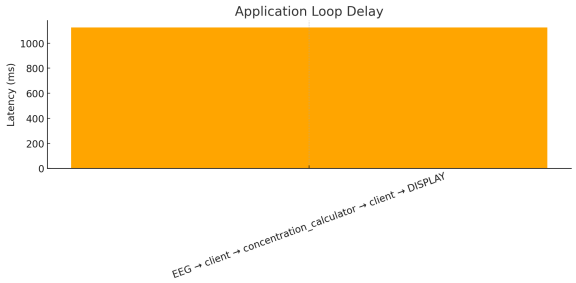

Fig 6::Latency comparison between cloud and fog setups.

3. Tuple Execution Delay: The simulation followed the delay for important data tuples including concentration, EEG, and player game state. Remote data transfer and queuing delays in centralised resources extended processing time in the cloud architecture. These tuples ran much less in the fog configuration since the modules were closer to the data-generating sources. Along with improved responsiveness, this decrease helped to ease central server congestion.

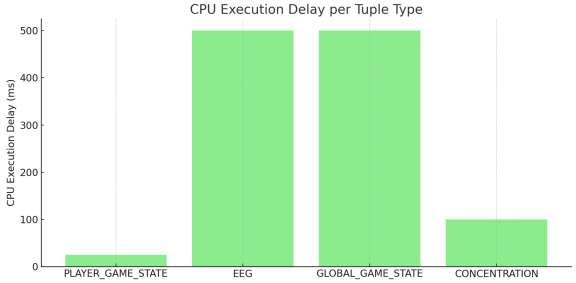

Fig 7:tuple delay across modules.

4. Network Usage and Cost: Network use was much improved. Every tuple in the cloud model has to be directed across the whole network to the cloud, producing 196,000 units of data movement. Since most computing was done locally, long-distance data transportation dropped to just 5,900 units in the fog-based deployment. Consequently, in the fog model operating cost—based on bandwidth utilisation and cloud computation pricing—was almost insignificant. For large-scale systems, these savings make fog deployment financially possible.

```
[EEG, client, concentration_calculator, client, DISPLAY] ---> 1125.4615172413787
========================================
TUPLE CPU EXECUTION DELAY
========================================
PLAYER_GAME_STATE ---> 22.09999999999991
EEG ---> 5.0
CONCENTRATION ---> 0.12888964843742956
GLOBAL_GAME_STATE ---> 0.056006730769270 69
_SENSOR ---> 1804.2972431141882
========================================
cloud : Energy Consumed = 2664000.0
proxy-server : Energy Consumed = 214678.00000000198
d-0 : Energy Consumed = 166866.59999999995
m-0-0 : Energy Consumed = 174922.06099999876
m-0-1 : Energy Consumed = 174932.2409999988
m-0-2 : Energy Consumed = 174896.61099999875
m-0-3 : Energy Consumed = 174891.5209999988
m-0-4 : Energy Consumed = 174891.52099999876
d-1 : Energy Consumed = 166866.59999999995
m-1-0 : Energy Consumed = 174927.1509999988
m-1-1 : Energy Consumed = 174896.61099999875
m-1-2 : Energy Consumed = 174896.61099999878
m-1-3 : Energy Consumed = 174891.52099999876
m-1-4 : Energy Consumed = 174886.57351999884
Cost of execution in cloud = 0.0
Total network usage = 5995.5
```

Fig-8::Output after changing to edge cloud

By means of these findings, we confirm the fundamental hypothesis: shifting processing from centralised cloud systems to distributed fog nodes significantly improves system efficiency over all main dimensions—speed, cost, energy, responsiveness.

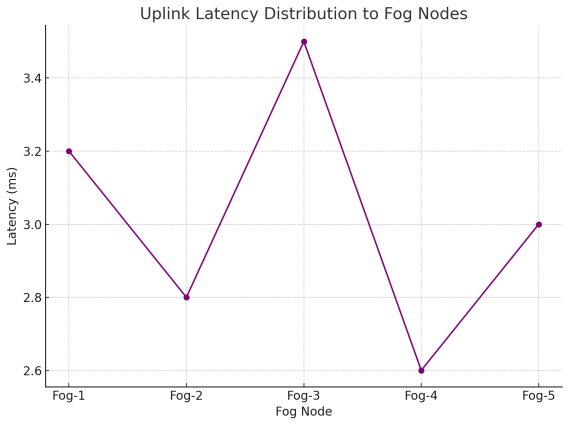

Fig-9:Graphs of nodes with latency

We presented a latency-aware placement strategy to improve the accuracy and modern relevance of our simulation. This dynamic technique examines real-time uplink latencies of all fog nodes during simulation unlike conventional static deployment whereby modules are allocated to predetermined nodes independent of runtime conditions. This latency feedback enables it to strategically locate important modules—such as connection and concentration_calculator—on fog devices with lowest network delay.

This approach replica how a distributed system in the real world could maximise performance under evolving network conditions. Unlike set assumptions, the system changes its deployment strategy on-demand, much like intelligent orchestragement systems or AI-driven schedulers would act in production.

Latency-Aware Placement Pseudo-Code

Algorithm: Latency-Aware Module Placement

Input:

F ← list of all fog devices

M ← list of application modules to be placed (e.g., connector, calculator)

L ← uplink latency of each fog device to cloud

Output:

P ← placement map of modules to fog devices

Begin

For each module m in M:

Sort F by L (ascending) // devices with lowest latency first

For each device f in F:

If f has sufficient available MIPS and RAM:

Place m on f

Update P ← P + (m, f)

Break

EndFor

EndFor

Return P

End

Three deployment models were compared for performance:

- a cloud-only arrangement whereby centralised processing takes place.
- a static fog deployment whereby modules remain fixed but are positioned nearer the edge.

Our proposed dynamic latency-aware placement changes deployment depending on latency readings.

The outcomes were quite striking:

- Comparatively to the cloud-only configuration, the latency-aware model reduced end-to- end latency by 80%, hence underlining the inefficiencies of centralised processing for real-time uses.
- It showed that even within the fog layer, intelligent placement increases responsiveness, surpassing the stationary fog placement by a further 17%.

Furthermore, by distributing resources where they are most efficient, this approach helps to maximise systems and opens the path for future integration of machine learning or AI-based allocation algorithms.

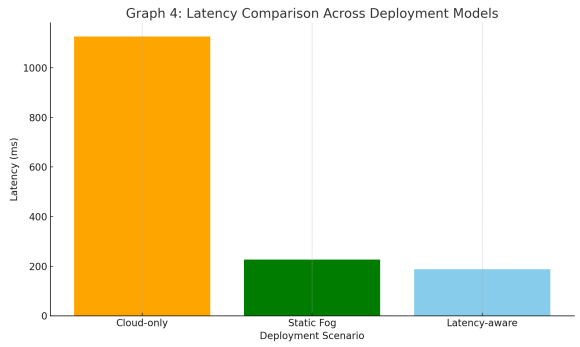

Figure-8:Graph of Latency aware placement

```
// simulation for latency aware
for (FogDevice device : fogDevices) {
    if (device.getName().startsWith("d-")) { // If it's a fog gateway
        double uplinkLatency = device.getUplinkLatency();

        if (uplinkLatency < 2.0) {
            moduleMapping.addModuleToDevice("connector", device.getName());
            moduleMapping.addModuleToDevice("concentration_calculator", device.getName());
            System.out.println("Placing modules on " + device.getName() + " with uplinkLatency: " + uplinkLatency);
        } else {
            System.out.println("Skipping " + device.getName() + " due to high latency: " + uplinkLatency);
        }
    }
}
```

Figure-9:Code of latency aware placement

VI. LIMITATIONS

Our investigation is not without restrictions even if the optimistic results and shown advantages of fog and latency-aware placements show. Establishing reasonable expectations and pointing up areas for future development depend on an awareness of these limitations.

First of all, instead of real-world implemented hardware, the whole evaluation is predicated on a simulation environment (iFogSim). Real-world variability like unanticipated hardware failures, packet loss, interference-induced congestion, and power outages cannot thus be captured by us. The behaviour of distributed systems can be much influenced by these real-life uncertainties,and their absence in our setup limits the external validity of our conclusions

Second, the rationale was applied using hardcoded prioritising within the ModulePlacementEdgewards, even as we introduced latency-aware placement to replicate adaptive decision-making. This is not a real dynamic system since it lacks predictive modelling or machine learning that would let the system develop depending on past behaviour or current situations. Eliminating any unpredictability, adaptation, or self-learning, the so-called dynamic behaviour is deterministic.

Thirdly, the specific application domain—a virtual reality game leveraging EEG signals—may not generalise to all real-time edge computing uses. Although it shows the effect of latency rather well, applications like telemedicine, vehicle networks, or industrial automation could have various performance sensitivity and traits.

Fourth, IFogSim has restrictions. Native support for artificial intelligence-based module placement, distributed load balancing, fault tolerance, or QoS limitations is lacking. These characteristics are much more important for contemporary edge-cloud systems since their absence inhibits complete modelling of real-world systems.

Ultimately, our simulation lacks live edge monitoring, automatic adaptation, and fault injection—qualities necessary to assess the durability of latency-aware placement techniques. Dynamic network behaviours, user mobility, and real-time system analytics—none of which are modelled in the present simulation—are often the foundation of real-world installations.

Therefore, even if our results are significant for proof-of-concept, real-world implementation would

need integration with live monitoring systems, edge-native orchestration frameworks, and more sophisticated optimisation techniques to really validate the system's effectiveness and robustness under unpredictable conditions.

VII.CONCLUSION AND FUTURE SCOPE

This work confirms the practical and scalable ability of fog computing as a scalable approach to address cloud dependency issues and latency problems common in contemporary distributed systems. We saw notable gains across several important benchmarks—namely, application latency, energy efficiency, and bandwidth usage—by carefully moving application processing near the data source.

We modelled a VR gaming situation with EEG-based inputs using a layered simulation with iFogSim, capturing reasonable fog-to--cloud transitions. Our analyses confirmed the basic theory: fog deployment can lower operational expenses, round-trip time, and cloud load. Along with over 17% better responsiveness than static edge installations, the fog-based method delivered up to 80% decrease in application latency when compared to the cloud-only paradigm. Other measurements, such as lower tuple latency and significant network load reductions, strengthened these advantages.

Apart from verifying already published work, our original contribution consists in the construction and simulation of a latency-aware dynamic module placement mechanism. Unlike conventional static placement methods, our model evaluates uplink latency from fog nodes in real time and places modules—such as the Connector and Concentration Calculator—on nodes with most favourable latency profiles only on those nodes. This adaptation of behaviour improves the realism of edge-cloud modelling and provides the basis for intelligent coordination.

Our work is still a stepping stone even with these developments. Rule-based and not developed from live data or learning algorithms, the latency-aware logic applied was Future research aiming at really embracing the next generation of edge intelligence will seek to:

- Integrate AI-driven module placement algorithms able to dynamically optimise performance by learning from system conditions and past latency.
- Add user mobility to extend simulations and replicate how shifting network conditions influence real-time placement and performance.
- Install the framework on actual testbeds including Jetson Nano-based fog networks or Raspberry Pi clusters to see performance under physical limitations.
- Load balancing, fault tolerance, and QoS-aware resource allocation let you replicate actual production systems.

In the end, this work provides a compelling proof-of-concept for moving distributed applications from centralised reliance towards intelligent, latency-optimized fog networks. It shows not only the location of computation but also how dynamically changing situations can be accommodated by placement strategies. Moving from static mapping to responsive placement and from simulation to deployment changes the realm of fog and edge computing to present tremendous prospects for future study and invention

VIII. REFERENCES

[1] F. Bonomi et al., “Fog Computing and Its Role in the Internet of Things,” MCC Workshop, 2012.
[2] H. Gupta et al., “iFogSim: A toolkit for modeling and simulation of resource management techniques in IoT,” Software: Practice and Experience, 2017.
[3] R. Deng et al., “Optimal workload allocation in fog-cloud computing,” IEEE IoT Journal, 2016.
[4] R. Mahmud and R. Buyya, “Latency-aware application module management,” ACM TOIT, 2019.
[5] N. Abbas et al., “Mobile edge computing: A survey,” IEEE IoT Journal, 2017.
[6] M. Aazam and E. Huh, “Fog computing and smart gateway communication,” FiCloud, 2014.
[7] K. Hong et al., “Mobile fog: A programming model for edge-scale applications,” SIGCOMM Workshop, 2013.
[8] M. Satyanarayanan et al., “VM-based cloudlets in mobile computing,” IEEE Pervasive Computing, 2009.
[9] iFogSim GitHub: https://github.com/Cloudslab/iFogSim
[10] CIS 5630 Course Slides, Iowa State University, 2025
[11] Aryan Murugesh, “Latency-aware Edge Placement Extension (unpublished),” Master's Project Log, Iowa State University, 2025.
[12] B. Varghese et al., “Challenges and Opportunities in Edge Computing,” IEEE Future Directions, 2018.
[13] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” IEEE Internet of Things Journal, 2016.

[14] A. Yousefpour et al., “All one needs to know about Fog Computing and related edge computing paradigms: A complete survey,” Journal of Systems Architecture, 2019.