ESP IJAST
ESP International Journal of Advancements in Science & Technology
ISSN: 2583-9233/ Volume 4 Issue 3 July-September 2026
Paper Id: IJAST-V4I3P103/Page No: 34-47

Original Article

A Lightweight Decision Framework for Choosing Between Edge and Cloud Processing in Latency-Sensitive Applications

Rahul Verma
Independent Researcher, India

Received Date: 30 July 2026 Revised Date: 05 September 2026 Accepted Date: 10 September 2026

Abstract: *Latency sensitive applications demand fast processing and timely response to ensure correct, reliable and efficient operation of the application. Autonomous Vehicles, Industrial Automation, Health Monitoring Systems, Smart Cities, Interactive Systems etc. usually require strict response times. Traditional cloud computing offers scalable processing power and central control; however, sending data to remote servers introduces delay in transmission (network delay), congestion, and communication overhead. Edge Computing reduces delays by processing information near the source of data, however edge devices are restricted in terms of computing power, memory, and power. This paper proposes a light-weight decision support system to help determine whether to process on the Edge or Cloud based on the requirements of the application and the operational environment. The system will evaluate the following parameters: Response Time Requirements, Network Delay, Bandwidth Available, Computational Complexity, Power Usage, Data Privacy, and Resource Availability. The system will use a straightforward decision process to determine where to perform the processing of data and does not require complex optimization techniques. The proposed approach will also allow for dynamic decision making based on changes in network environments and/or application workloads. With the evaluation of several operational parameters, the proposed approach intends to increase response time reliability, resource utilization, and processing efficiency. This paper creates a theoretical base for the development of real world decision support systems for latency sensitive applications. System designers will find value in this decision support system as it assists them in balancing responsiveness, scalability, security, and resource constraints when deciding if computational tasks should be performed at the edge or remotely in the cloud.*

Keywords: *Edge Computing, Cloud Computing, Latency-Sensitive Applications, Decision Framework, Response Time, Network Latency, Resource Allocation, Energy Efficiency, Data Privacy, Computational Offloading*

I. INTRODUCTION

The exponential growth in use of connected devices, AI and other high-volume digital service applications has increased the need for fast, reliable and contextual-based computer systems. Many of the applications that are developed using these types of technologies (e.g., automated vehicles, intelligent healthcare systems, industrial automation, intelligent surveillance systems and interactive digital service applications) often have tight latency requirements. A delay in the processing of information could lead to decreased service quality, interruption in operations and/or poor decision making regarding safety-related issues. For many years, cloud computing has provided a viable solution to support these applications through centralized processing, large-capacity storage and scalable computing resources. However, when sending data from devices to cloud servers located far away and then sending the processed data back to the device, there can be communication latency, network congestion and increased power usage. To address some of the issues associated with cloud computing, edge computing allows for local processing of data generated by devices; this reduces the amount of time required for data transmission, as well as dependence upon remote infrastructure. Edge devices, however, typically do not have the same level of capability as centralized cloud platforms, therefore selecting which platform is best suited for each application remains a significant design concern.

The selection of either edge or cloud processing goes beyond just selecting the option which will have the least amount of latency. The types of applications that exist each have their own unique requirements for how much computing power they will need, the amount of data they will generate, what level of security is required for them, and how quickly they will expect to receive results. For example, a smart traffic control system may need to use an edge-based solution to immediately recognize vehicles and other road hazards, while a city-wide analysis of historical traffic patterns could benefit from using a cloud-based solution. Similarly, a wearable device that monitors a person's health may need to make immediate decisions based on urgent events happening in real-time on the device itself, while analyzing long-term medical data may require more computing power than what is available on the wearable device. Research has been conducted on offloading computation to various locations

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

(edge, cloud) by considering many variables including: communication delay; processing time; energy consumption; how the workload is distributed; and what network resources are available. While some researchers have developed complex optimization algorithms or reinforcement learning techniques to allow for dynamic assignment of tasks to the edge or cloud, these methods create additional complexity when implementing them, as well as create additional computational costs. Therefore, a lightweight decision framework may provide a simpler way for systems that need to quickly and efficiently assign tasks to either the edge or cloud, while making it easier to understand why certain decisions were made. A lightweight decision framework should evaluate operational conditions of the system through the use of simple rules, thresholds, or weighted criteria, so that tasks can be assigned to either the edge or cloud without having to train the system extensively or run costly optimization processes.

This paper presents a simple decision model to help determine whether to process data locally at the edge or remotely in the cloud based upon several factors such as the application's required response time, the anticipated network latency, the amount of processing required by the application, how much bandwidth is available to transmit the data from the device to the remote location, the limitations on power/energy consumption of the devices, the privacy concerns of the users/applications, and the availability of computing power on-site. The goal of this framework is to provide a user-friendly tool to assist developers/operators in deciding which type of processing (edge vs. cloud) would be better suited for an application depending on the specific needs of the application. For example, if an application requires immediate response times, has poor/no connectivity to other locations/networks, or needs to protect its user data locally, then edge processing may be a better choice. On the other hand, if an application requires large amounts of processing capacity and is willing to accept some level of latency/delay due to transmitting data across a network, then cloud processing may be a better choice. Additionally, this framework will allow users/operators to continually evaluate their decisions regarding processing locations based on changes in operating conditions (i.e., increased workload, reduced network performance, decreased local resource availability). This conceptual framework needs to be validated empirically before we can confirm that it provides performance advantages. Ultimately, this framework intends to provide a simple yet effective decision-making method that can be implemented with minimal overhead and easily customized for various application environments. Additionally, this work establishes a base for subsequent experimental evaluations of the framework utilizing representative workloads, measurable indicators of latency, energy consumption, and deadline satisfaction. In summary, the primary objective of this work is to establish a simpler, more responsive, and practical approach to managing resources in order to enable greater efficiency when deploying edge-cloud computing solutions for applications that require fast/real-time processing.

II. PROPOSED LIGHTWEIGHT DECISION FRAMEWORK FOR EDGE AND CLOUD PROCESSING

The proposed lightweight decision-making process will provide a structured approach to make decisions about which tasks (tasks that have time constraints) can be processed on the "edge" versus the "cloud." This process will assess the critical needs of an application prior to making a choice regarding where to process. A few of the key inputs used to make this determination include the maximum amount of time that can be tolerated for a response, the estimated amount of time it takes to transmit information over the network, the size of the workload being computed, the amount of bandwidth available to the system, the amount of energy required to run the application, the need to protect sensitive information, and the total amount of computing capability available in the system. The use of these variables will allow us to determine which environment (edge vs. cloud) would best suit the needs of the application. If the application requires a quick response, processes data locally, or has limited network connectivity, then processing on the edge would be the preferred option. However, if the application has high computing demands and network latency is within acceptable levels, then processing in the cloud would be the preferred option. Since we are using simple decision rules, we will avoid the additional complexity and expense associated with using complex optimization techniques.

The first step of the decision-making process is to collect all of the information about what needs to be done (task requirements) and how things are working at this moment (current operating condition). The system then calculates an estimated amount of time required to complete each type of processing alternative. When calculating the estimated amount of time to process something on-site, the calculation includes the time it takes to do the processing locally plus any additional communication costs associated with doing the processing on-site. When calculating the estimated amount of time to process something in the cloud, the calculation includes the time it takes to send the data to the cloud, the time it takes for the data to travel through the network to get to the cloud, the time it takes to process the data in the cloud, and the time it takes to deliver the results back to the user. After the system has calculated estimates for each of the two types of processing alternatives, the system compares those estimates to the amount of time the user wants the system to take to complete the task. If cloud-based

processing will not allow the system to complete the task within the user's desired amount of time, then the system chooses to perform the task on-site if there are enough on-site resources to do so. If neither type of processing alternative will allow the system to complete the task within the user's desired amount of time, the system may choose to use the closest possible alternative, decrease the user's workload, or inform the user that their request cannot be completed at this time. Restrictions on privacy and available resources are viewed as decision constraints rather than as options that can be ignored.

The framework includes methods to adjust to changing network and workload conditions. If the network becomes congested, it might be more effective to execute at the cloud; if the local workload is too high, there may be fewer advantages to executing at the edge. At regular intervals, the system will assess these factors and respond to major changes in operational conditions. In order to avoid excessive changes in processing location, the system can use a minimum switching interval (or a threshold) to limit the frequency of location assignments. Limiting the number of location assignment changes minimizes the amount of communication required to synchronize data across nodes and helps stabilize processing decisions. Depending on the needs of the application, the framework can implement its decision making logic using conditional rules or a weighted scoring mechanism. Safety critical applications must have alternative options available for executing tasks, and they must not rely solely on estimated latency to determine operational decisions.

Table 1 identifies the most significant decision parameters that will influence the selection of the processing site. They offer a concrete way of assessing the performance of the proposed model in various application scenarios. This model is suitable for monitoring smart cities, controlling industrial processes, monitoring patients with remote devices (connected health care), controlling traffic flow (intelligent transportation) and other types of latency-sensitive systems. The primary benefit of this model is its simplicity, as it uses understandable decision criteria combined with several operational constraints. However, the model's performance must be verified experimentally in terms of how well it works under different network conditions, different levels of workload, and different hardware configurations. Measurements of response time, adherence to deadlines, power consumption, completed tasks, and decision-making cost will enable us to determine if the model produces significant improvements over off-the-shelf edge-only, cloud-only, and alternative offloading strategies.

Table 1: Key Factors in the Edge–Cloud Decision Framework

Factor	Edge Processing	Cloud Processing	Decision Criterion	Measurement	Purpose
Response time	Usually lower local delay	May include network delay	Application deadline	Milliseconds	Ensure timely results
Network latency	Limited remote communication	Depends on connectivity	Estimated transmission delay	Milliseconds	Assess communication overhead
Computational workload	Suited to manageable tasks	Suited to resource-intensive tasks	Processing demand	CPU cycles or execution time	Match task complexity
Bandwidth availability	Can reduce data transfers	May require substantial data transfer	Available bandwidth	Mbps	Avoid network congestion
Energy consumption	Depends on device capacity	Includes communication and remote execution costs	Energy budget	Joules	Improve energy efficiency
Data privacy	Supports local data processing	May require external data transfer	Privacy restrictions	Policy compliance	Protect sensitive information
Resource availability	Limited local capacity	Potentially scalable resources	Available CPU and memory	CPU usage and MB	Prevent resource overload
Workload variation	May become overloaded	Can absorb additional workloads	Current system utilization	Percentage	Support adaptive decisions

III. OPERATIONAL METHODOLOGY AND DECISION-MAKING PROCESS

The operating approach of the suggested lightweight decision-making method focuses on identifying a suitable processing location based on the applications requirements and the systems present state. First, the application identifies what is needed

computationally for the application to be completed, what size of input data is required, how long the application must take to finish, if there are any privacy concerns regarding the application, and approximately how often the application will run. All this information is obtained either directly from the application or by using a light-weight monitoring mechanism. Next, the method gets information about the current status of edge devices and cloud connections such as; how much processing power is still available, how much memory is being used, how much time it takes to transmit information back and forth across the network, and how fast information can travel over the network. Using the above-mentioned pieces of information, the method then determines if both options for processing (edge vs. cloud) are able to meet the applications minimal requirements. By performing this preliminary evaluation, the method eliminates unnecessary transfer of tasks from one option to another and creates a systematic way to make decisions about where to process a given task. Since the method does not require large amounts of training data or complex optimization algorithms, it may be used effectively in environments with limited resources.

The next step of our methodology is to estimate the total response time for each option of edge or cloud execution. Total response time is composed of two main components: edge response time and cloud response time. Total response time at the edge is primarily made up of local processing and any communication needed within the edge environment. Cloud response time includes transmitting data from the local platform to a remote platform, network communication delays during transmission, cloud-side execution of the task, and returning results back to the user. The methodology compares both types of estimated response times against the maximum allowed response time of the application. If the edge option can complete the task in the required amount of time, but the cloud option will cause too much delay due to communication issues, then using the edge option is usually the better choice. On the other hand, if the task requires more computing power than what is available at the edge, and the estimated communication delay is reasonable, then using the cloud option is typically the best choice. It is important to remember that all estimated response times should take into account measurement errors and variability in network performance. Additionally, for applications that have rigid deadlines, the methodology can provide more conservative latency estimates instead of relying solely on average response times. By providing more conservative estimates, the methodology allows users to make more reliable decisions as operating conditions change.

Third, the evaluation of other restrictions that could affect whether the processing location would work for the task. Examples of such restrictions include data privacy requirements (which could limit where sensitive data can leave its local environment), energy restrictions (which could limit the length of time a device can perform calculations if it runs out of battery power), and resource availability (which could limit how well an edge device performs if it has high CPU usage or low memory). Therefore, the framework applies required restrictions prior to applying optional performance options. When a processing location does not meet the required restrictions for the task, the location is removed as an option if there is another location that does meet the required restrictions. After removing the locations that do not meet the required restrictions, then a simple weighted scoring system can be used to compare the remaining options based on latency, energy use, bandwidth use, and resource use. The weights can be modified based on application priorities. For example, industrial control systems would likely place more emphasis on response time; whereas battery-powered monitoring devices would likely emphasize energy conservation. By allowing for flexibility in prioritizing the criteria, the framework is flexible enough to accommodate different applications without having to develop a complex mathematical optimization model.

The last phase is called execution, monitoring of the performance, and then re-evaluating on a regular basis. Once the location for the processing has been chosen, the task is assigned to that location; and the operational parameters that are monitored include how long it took to complete the task, did the completion meet the required deadline, what amount of energy was consumed, and what amount of data was transferred over the network during the process. These metrics can then be compared to what was predicted as far as performance goes and the thresholds for making decisions about which location would be best can be adjusted based upon these comparisons. If network congestion increases, an edge device is being over-utilized, or the application's work load has changed significantly, the framework will re-evaluate the available options. In order to minimize the number of times that decisions are made to switch between edge and cloud processing, the re-evaluation of the available options should only occur after there is a significant change in the operating environment, not after every minor change. Additionally, a minimum amount of time required between switching decisions or a hysteresis value used to determine when a decision should be made can assist in creating stability in the decision making process. If none of the available options (edge or cloud) will allow for satisfying all of the requirements of the task, then the system should provide notification of the inability to meet the task requirements and implement an appropriate fall-back strategy for the application. The suggested methodology provides a workable foundation for testing and evaluating its efficacy in comparison to alternative processing methods with

similar workloads, network conditions, and hardware components to assess its effects on latency, energy usage, utilization of resources, and reliability.

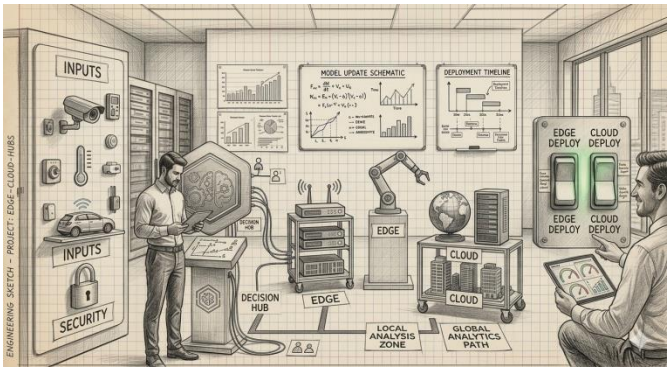


Figure 1: Operational decision-making flowchart for edge-cloud workload allocation

IV. IMPLEMENTATION AND EVALUATION OF THE LIGHTWEIGHT DECISION FRAMEWORK

A. Framework Architecture and System Components

The lightweight decision-making framework being proposed has five primary components; Task Monitoring, Resource Assessment, Latency Estimation, Decision Selection and Execution Management. Task monitoring component identifies the application requirements, such as computing complexity, size of input data, response time limits and privacy requirements. Resource assessment component determines the available processing capacity of edge devices and cloud computing. It takes into consideration CPU usage, memory usage, network bandwidth and device power limitations. These measurements allow the system to determine if a task can be processed in an efficient manner in either environment. The system design was created with low communications between each component to reduce unnecessary processing. Each component performs its own function, therefore keeping the system modular and adaptable to various types of applications.

The latency estimation component is responsible for determining the approximate amount of time it takes to process a request (on either the edge or the cloud), including all aspects of processing including but not limited to; local processing, transmitting data, network delays, and/or remote processing. The decision-making component uses either a set of pre-defined criteria or a simple scoring model to determine which processing location is best suited for each request. Requirements that are mandatory (e.g., privacy requirements and/or meeting response deadlines) will always be evaluated prior to making any optional performance evaluations. Once a processing location has been chosen, the execution management component will assign the request to that location and continue to monitor the performance of that request. If network conditions or resource availability significantly alter during the processing period, the framework may re-evaluate the initial processing decision. While this architecture allows for flexibility in adjusting to changing conditions, the overall success of the architecture depends heavily upon accurate monitoring, reliable estimates of latency, and proper threshold levels for decision-making.

B. Performance Evaluation Metrics and Experimental Design

The proposed framework can be evaluated by creating a controlled experimental environment representing various latency-sensitive workload types. Examples of suitable test cases include smart-city sensor processing, industrial monitoring, healthcare event detection, and connected transportation applications. Each test case will consist of several tasks having different computational needs, data size, response time limit, and privacy requirements. The proposed framework can be deployed on edge devices linked to a cloud server via networks with adjustable latency and bandwidth. Edge-only processing, cloud-only processing, and a suitable baseline offloading strategy should all be compared to the proposed framework. The methods compared to the proposed framework should be subjected to the same workloads and operated in the same experimental condition to allow for valid comparisons.

To evaluate the effectiveness of the proposed framework, it will be essential to utilize multiple performance measures. An average response time will indicate the average amount of time required for completing a task; on the other hand, the 95th percentile response time will identify delays associated with tasks that are slower than average. A deadline satisfaction metric will provide an indication of what percentage of tasks are successfully completed within their specified response times. Additionally, energy consumption metric will measure the amount of energy consumed by devices during their operation as well as during any communication activities that occur. Furthermore, a resource utilization metric will assess how efficiently the

available processing capabilities of each device are being utilized. Finally, a decision overhead metric will provide an indication of how much time and computational resources are required when selecting a processing location for each task. In addition to providing indications of system reliability under varying network conditions, the task completion rate metric will also provide insight into the overall reliability of the proposed framework. To minimize the effects of random variations in data, it is recommended that multiple repetitions of each experiment be carried out. Tables and graphs can then be used to summarize the experimental results, and if appropriate, a statistical analysis can also be performed. As this research is intended to develop a conceptual framework, only after experiments have actually been conducted should any quantitative improvements in performance be reported.

C. Adaptive Decision Rules and Practical Deployment

Adaptive decision rules allow the framework to adjust to different applications' needs and changing operational conditions. First, the framework determines if each processing location meets the application's required constraints. Then, if the application has a short-term need to run a task on an edge device and there is enough edge capacity, edge processing will be given higher priority. However, if the application can wait to have its task processed by a cloud service and has no problem waiting for the results, then cloud processing would be a good option. In cases where both options meet the application's requirements, the framework can weigh several factors including response time, energy consumption, bandwidth consumption and available resources. The weights used should depend upon what is important to the application itself and not be the same for every type of environment. An example of this would be that an industrial control application might want to maximize its ability to provide predictable response times, whereas a large scale analysis application might be more interested in increasing its processing capability and scalability.

Practical deployment must also have ways of dealing with inaccurate measurements, intermittent loss of connectivity, and unexpected shortages in resources. It must be able to periodically estimate performance levels and establish limits so it does not switch back and forth from edge to cloud execution too often. If tasks can be divided into smaller pieces, the workload can be split between both the edge and the cloud as long as the cost of communication does not cause the time to complete the task to exceed the maximum allowed latency. When data contains sensitive information and the policy does not allow for transmitting this information outside of the device, then the data must stay on the device. When safety-critical applications are being run, the framework should use previously defined backup procedures and never assume that there is another location where the processing can take place. Deployment testing must evaluate how scalable, interoperable, secure, and reliable the system performs across various hardware platforms and network setups. These aspects are critical for transitioning from a conceptual decision-making process to an operational system. Future studies can determine if light-weight decision-making processes provide any significant advantages over current methods used to decide which processing option (edge or cloud) to utilize when running real-world applications.

V. BENEFITS, CHALLENGES, AND PRACTICAL APPLICATIONS OF THE PROPOSED FRAMEWORK

The proposed lightweight decision framework provides a realistic solution for choosing between edge and cloud processing in applications that are sensitive to delays (latency). The primary advantage of the proposed lightweight decision framework is that it can make processing choices based upon a small number of operating parameters and simple decision criteria. The framework will consider response time requirements, network conditions, computational demands and resource availability; therefore, the framework can minimize unnecessary data transfers and provide better allocation of tasks. Edge processing enables fast response times and processes data locally, whereas cloud processing allows users to obtain additional computational resources. In addition to being able to assist in making processing choices, the lightweight decision framework can also adjust its processing choices when workloads or connectivity states change. Some advantages of utilizing the proposed lightweight decision framework include enhanced responsiveness, improved resource utilization, decreased communication overhead, and increased flexibility in various heterogeneous computing environments. However, the above-mentioned advantages are contingent on obtaining reliable measurement information, selecting appropriate decision thresholds, and on the type of application being deployed.

While there are several benefits of using this framework, there are also numerous issues that need to be addressed. Network latency and available bandwidth are constantly changing; therefore, predicting response times becomes very difficult. Edge devices can be overloaded, and the connection to the cloud can be lost or severely degraded. When tasks are moved from one processing location to another, additional communication costs can be incurred, state management issues can arise, and security concerns can emerge. Therefore, the framework should attempt to limit how often it switches from one processing

location to another, and it should provide suitable fall-back mechanisms when both processing locations do not meet the specified requirements. Some examples of where this framework can be used in practice include smart city traffic monitoring, industrial automation, connected healthcare, intelligent transportation systems, and real-time surveillance. For instance, a traffic monitoring system may use local processing for high priority vehicle detection events and aggregate data for long-term analysis by sending aggregated information to the cloud. Future studies should test the performance of the proposed framework in the context of these applications using experimental methods. In addition to testing against edge-only, cloud-only, and other existing offloading strategies, the performance of the proposed framework should be evaluated in terms of response time, deadline satisfaction, energy consumption, resource utilization, and decision overhead. This type of testing is required prior to providing quantitative evidence supporting the performance enhancements or operational advantages associated with the use of the proposed framework.

VI. COMPARATIVE ASSESSMENT AND OPTIMIZATION OF EDGE–CLOUD PROCESSING

A. Assessment of Processing Strategies and Decision Factors

The selection of a processing method (edge vs. cloud) will be based on the characteristics of the applications being processed; specifically, the time in which they need to provide a response, the amount of computing power required to complete the task, the state of the network connections available at that point in time, the power limitations of the device(s), and the level of security/privacy needed for the information. In general, applications that require quick turnarounds or contain highly confidential/sensitive information that must be kept locally would benefit from edge processing. On the other hand, applications that require large amounts of computing power, have centralized data storage needs, or perform analysis on a scale that cannot be done on a single node would likely be better served by cloud-based processing. Neither of the two approaches is inherently "better" than the other. To determine the most appropriate approach for each individual application, the decision-making process outlined in this paper evaluates the characteristics of the application relative to the capabilities of both edge- and cloud-based environments. This process determines if either of the two environments can meet the minimum operational requirements of the application, and then selects the most appropriate alternative based on the relative performance of the alternatives. This process reduces the likelihood of incorrectly allocating tasks to a particular type of processing environment, and creates a consistent framework for evaluating the performance of different types of applications running across different types of computing environments. Table 1 outlines some of the major differences between the two types of processing environments and how those differences impact the decision-making process.

Table 2: Comparative Evaluation of Edge and Cloud Processing Based on Performance, Resource Requirements, Privacy, and Deployment Considerations

Evaluation Factor	Edge Processing	Cloud Processing	Decision Implication
Response time	Usually lower communication delay	Depends on network and remote execution	Prefer edge for strict deadlines
Computational capacity	Limited by local hardware	Generally scalable	Prefer cloud for intensive workloads
Bandwidth demand	Can reduce external data transfers	May require substantial transmission	Consider data size and connectivity
Energy requirements	Uses local device resources	Includes communication and remote execution costs	Evaluate total energy consumption
Data privacy	Supports local data retention	May require external data transfer	Apply privacy restrictions
Scalability	Limited by distributed device capacity	Supports centralized resource expansion	Consider workload growth

The evaluation must rely upon quantifiable metrics as opposed to assuming which type of environment (cloud or edge) is superior. An example would be a task that has a very short response time; however, if the network is fast enough and the edge device is currently running at maximum capacity, then the task could potentially run in the cloud. Another example would be a task that requires extensive computation and there is no available means of communication with external systems, and there is a suitable edge device, then the task would most likely have to execute on the edge device. Therefore, the assessment framework must consider the total estimated response time, assess the availability of resources and potential privacy issues prior to

determining where a task will be executed. If both environments are capable of fulfilling the requirements, then a simple scoring system can be developed to rank the two options based upon the priority level of the application. A scoring system like this will increase the transparency of the process by allowing the user to understand why each decision was made. However, the accuracy of the evaluation process depends on accurate measurement devices, appropriate threshold levels, and regular calibration of the system as operating conditions change.

A. Optimization Strategies for Reliable and Efficient Processing

The goal of enhancing the effectiveness of the proposed architecture is to increase the precision of making decisions, while at the same time decreasing the cost of computing resources. One approach to achieve this objective is to utilize adaptive threshold values based on the current measured network latency, the load placed upon the edge devices, and the deadline of applications. The proposed architecture will be able to adjust these values in real-time as a result of performance measurements that have occurred in close proximity to the time of adjustment. This enables the architecture to react to changes in connectivity and/or processing requirements without having to run a complex optimization algorithm. A second strategy to accomplish the same goal is to implement a weighted scoring methodology for the alternative solutions that exist within the set of feasible solutions. To do so, the scoring system would give more weight to those aspects of each solution which are more relevant to the specific application. For example, an industrial control system would probably value predictable response time more than it would value energy conservation and/or communication efficiency. Conversely, an environmental monitoring application would likely value energy conservation and/or communication efficiency more than it would value predictable response time. Additionally, the proposed architecture could incorporate hysteresis threshold values into the decision process. Hysteresis threshold values would serve two purposes. First, they would enable the architecture to avoid repeatedly changing its decision regarding where to place tasks when there is very little difference in performance between processing at the edge versus processing in the cloud. Second, they would reduce the amount of overhead associated with reassigning tasks to other locations when the difference in performance is minimal.

Further optimizations can be made using workload classification, selective data transmission, and resource-aware task scheduling. Task classifications include urgency, computational complexity, data sensitivity, and the ability to tolerate communication delay. Urgent tasks can be prioritized at the edge, and analytically complex workloads can be processed on cloud infrastructure. Only necessary information or aggregated results should be transmitted to reduce bandwidth usage and potential exposure of sensitive data when possible. The framework must monitor device usage such as CPU usage, available memory, and energy constraints to avoid overusing resource-constrained devices. For applications that have dependent tasks, scheduling decisions must consider how communication occurs between the various processing steps since transferring individual tasks may extend the amount of time it takes to complete all of the tasks. To determine how effective the optimizations are, repeated experiments should be performed with different types of workloads and different network conditions. Measures used to assess the effectiveness of the optimizations should include average and percentile response times, whether or not deadlines were met, energy consumed, task completion rate, and decision-making overhead. Future research can examine whether adaptive weighting and predicting latency provide better results than simpler rule-based decision-making strategies.

VII. FUTURE DIRECTIONS AND RESEARCH DEVELOPMENTS

A. Integration with Artificial Intelligence

The use of artificial intelligence in the lightweight decision framework can help determine which processing location is best suited for execution when there are continuous changes in workload and network conditions. Machine learning models can be used to evaluate previous execution times, resource utilization, and communication delays to assess whether an application will run better on the edge or in the cloud. Although computationally intensive machine learning models have the potential to introduce additional overhead and limit the usefulness of the decision framework on resource constrained devices; a more viable solution would be to implement light weight predictive models that supplement rule based decisions rather than completely replace them. These models would enable the identification of recurring workload patterns and enable recommendations regarding appropriate processing locations prior to degradation in performance. Future research needs to investigate whether these predictive capabilities lead to quantifiable advantages compared to traditional threshold based mechanisms. Assessments of the predictive capabilities should include evaluation criteria such as prediction accuracy, decision latency, energy consumption, and computational overhead.

B. Security and Privacy-Aware Processing

Security and privacy issues are crucial factors in deciding where to run applications, either at the edge or on a cloud platform. When using edge computing, you have to send less of your sensitive data to the cloud than if you were running your application on the cloud. However, if you use cloud computing, there is a risk that your sensitive data may be exposed as it is being sent from the edge device to the cloud platform, or as it is stored in one location on the cloud. Therefore, future versions of our proposed framework should consider privacy policies, authentication requirements, encryption overhead and other security-related processing constraints when making decisions about which application workload should be run at the edge or on the cloud. Some types of tasks that contain sensitive information should be limited to running only within authorized environments. On the other hand, non-sensitive computational workloads can be placed on the edge or on the cloud based upon their latency and resource needs. Our proposed framework must also consider how much extra processing time is required due to security measures. This is especially true for those applications that require real-time processing. Additionally, future research could examine how security requirements impact task placement under various types of threats and network configurations.

C. Application in Emerging Computing Environments

The proposed framework could also be used in emerging types of computer-based environments such as Internet of Things networks, connected vehicles/transportation, smart factories/manufacturing, healthcare monitoring systems, and intelligent cities. The variety of work generated in each type of environment will have different response time requirements, resource requirements, and communication patterns. For example, in smart manufacturing, there may be an urgent task to monitor the operation of machines that can only be done on-site; however, long-term analysis of data collected on previous production runs could be analyzed using cloud-based resources. Similarly, in connected vehicle/transportation systems there will be both immediate on-site event detection (e.g., accident) and centralized traffic analysis. Extending the framework to operate within all of these different environments will necessitate taking into account the heterogeneity of devices (i.e., different types of devices being utilized), occasional loss of network connectivity, interdependencies among tasks/workloads, and different service requirements. Further research should focus on developing specific configurations for each environment while maintaining a single decision-making structure. Experimental implementations of the framework would allow researchers to assess whether the framework can successfully adapt to different hardware platforms and communication technologies. Such research may lead to the development of a flexible and transferable method of selecting where processing should occur within distributed computing environments.

D. Experimental Validation and Scalability

In order to validate the practicality and effectiveness of the proposed lightweight decision-making framework through experimentation; it will be necessary to perform experiments on edge devices, cloud environments and configurable networks that are representative of real-world scenarios. Additionally, experiments should compare the framework to edge-only execution, cloud-only execution, and existing task offloading approaches, each executing the same workload. Evaluation metrics used in evaluating the performance of the various execution options should include average response time, ninety-fifth percentile latency, deadline satisfaction, energy consumption, resource utilization, and decision overhead. Experiments should evaluate how the decision-making framework performs when subjected to network congestion, increased workload intensity, limited edge capacity, and temporary connectivity failures. Furthermore, the ability of the decision-making framework to scale as the complexity of the system increases due to additional devices and concurrent tasks should be examined. Statistical analyses of repeated trials would provide greater confidence in the validity of the experimental findings. Ultimately, the results obtained from these experiments can assist in identifying potential limitations of the decision-making framework, provide data for parameter tuning, and determine if the decision-making framework provides benefits relative to other decision-making approaches in latency-sensitive applications.

VIII. DISCUSSION AND IMPLICATIONS OF THE PROPOSED FRAMEWORK

The purpose of the decision-making model as described above is to provide a user-friendly and manageable method for determining whether to process data at the edge of the network (edge) or at the cloud (cloud) when dealing with latency-sensitive applications. To accomplish this goal, the decision-making model will take into account several different factors including response time, processing power available, energy consumption, network performance and security requirements. The decision-making model is designed to operate as a simple decision-making process, rather than relying on complex optimization algorithms or centralized processing models. In addition to being simple to use, the decision-making model is also designed to operate under the assumption that there are limited computational capabilities available in the system. As such, the decision-making model is intended to be used in conjunction with other types of decision-making systems that are able to provide

additional levels of sophistication. The decision-making model is based upon two primary components: measurable indicators and predetermined decision rules. These two components work together to establish a baseline for making decisions regarding processing location. It should be noted that the effectiveness of the decision-making model is directly related to how well the data collected was measured, how appropriate the threshold values were established for each indicator, and how quickly the model can adapt to changes in workload demands.

The framework also provides the opportunity to better manage computational workloads in environments with varying levels of heterogeneity. For example, edge devices can perform short-term processing near where the data was collected; whereas, the cloud can provide additional processing power for long-running, resource-intensive computations. This separation of concerns enables applications to take advantage of the most appropriate resource for each type of computation, instead of having to direct all computations to a single computing platform. As examples, Smart City systems can process emergency traffic events locally and then send aggregated data to the cloud for longer term analysis. Industrial monitoring systems can likewise quickly identify faults at the edge and use cloud resources for historical analysis and predictive maintenance. However, when allocating tasks to different parts of the network (e.g., edge vs. cloud), it is critical to account for interdependencies among the stages of processing, communication costs, and potential for device overload at the edge.

The framework also emphasizes the need to balance performance and energy efficiency with security and data privacy. When transferring large quantities of information to cloud servers there is an increased risk of high bandwidth usage as well as high communication-related energy consumption. Local processing of sensitive information can decrease unnecessary exposure of data; however, local devices will still require appropriate security measures and adequate computing power. Conversely, executing tasks in the cloud can be beneficial if local hardware cannot accomplish a given task before the deadline. Therefore, the decision-making model proposed in this study treats both privacy restrictions and the availability of resources as key constraints while taking into account the latency and energy consumption associated with each alternative. It should be noted that in real world scenarios, some of these criteria may conflict with each other and therefore require application-specific priority setting. A lightweight scoring system can aid in ranking viable options; however, the weights used in such a system must be chosen wisely and tested in real-world scenarios.

The framework has some limitations that should be noted. The estimated response time could be different than what will occur in reality as a result of varying amounts of network congestion, changing cloud workloads and/or fluctuating usage of devices. There is also the possibility of introducing additional monitoring overhead if there are too many frequent assessments. Additionally, the frequent change in processing locations can lead to increased communication cost and make coordination of tasks more difficult. Finally, the proposed framework is just a concept until it can be tested experimentally. It cannot be assumed to outperform existing methods without systematic evaluation. Future studies should examine dynamic thresholding, prediction of latency, classification of workloads, and privacy aware scheduling. Experimental evaluations should include measuring distribution of response times, meeting deadlines, energy consumed, resources utilized and overhead of making decisions across various applications. This information would help clarify the conditions where the framework would provide significant operational improvements.

The main advantage of this proposed approach is that it provides an base-line to make cost-effective decisions regarding how to process data at the edge of the network and in the cloud. The primary benefit of this approach is that it incorporates all necessary processing constraints into a simple and understandable decision-making process that can be tailored to meet the needs of each application. This allows for local processing of time-critical applications and cloud-based processing for large scale applications that require significant amounts of computational power. In general, this approach will improve the way tasks are allocated across the network, decrease the amount of unnecessary communication between nodes, and provide users with more information about the availability of various types of computational power. To ensure successful implementation, careful attention must be given to the manner in which the proposed approach is implemented, monitored, and supported by appropriate fallback mechanisms. In addition, empirical testing should be conducted to evaluate the performance of the proposed approach. Further research should focus on evaluating the performance of the proposed approach on actual workloads and comparing the results obtained from such evaluation with those obtained from existing approaches using similar evaluation criteria. Such evaluation will help identify the feasibility of deploying the proposed approach in real-world situations and provide direction for enhancing its ability to support low latency applications.

IX. PERFORMANCE EVALUATION CRITERIA AND EXPECTED OUTCOMES

A critical aspect of evaluating the performance of the proposed lightweight decision-making process is to determine if it will meet the requirements of real-time applications. To do this, we need to assess how well the lightweight decision-making process can choose whether to process data on-edge or on-cloud based upon various factors such as workload, network status and available resources. We can create a controlled testbed environment by connecting on-edge devices to a cloud-based infrastructure via networks that have adjustable latencies and bandwidths. In order to compare the performance of the lightweight decision-making process with other processing methods, we will need to execute the same workload across each of these processing methods, while maintaining identical conditions for each experiment, so we can make an objective comparison. To assess the performance of each processing method, we will evaluate each method's ability to minimize response time; maximize deadline satisfaction; minimize energy consumed; maximize resource utilization; maximize task completion rates; and minimize decision-making overhead. All of these metrics collectively contribute to our overall understanding of processing efficiency and operational reliability.

Table 3: Performance Evaluation Metrics for Assessing the Efficiency, Reliability, and Resource Optimization of Edge–Cloud Computing Frameworks

Evaluation Metric	Measurement Method	Expected Objective	Significance
Average response time	Mean task completion time in milliseconds	Reduce processing delays	Measures overall responsiveness
Deadline satisfaction	Percentage of tasks completed within deadlines	Increase successful timely execution	Evaluates application requirements
Energy consumption	Energy used per task in joules	Avoid unnecessary energy expenditure	Supports energy-efficient computing
Resource utilization	CPU and memory utilization percentages	Balance processing workloads	Identifies resource bottlenecks
Task completion rate	Percentage of successfully completed tasks	Improve operational reliability	Measures processing success
Network overhead	Data transmitted and communication activity	Reduce unnecessary transfers	Evaluates bandwidth efficiency
Decision overhead	Time and resources required to select a location	Maintain lightweight operation	Measures framework efficiency
Latency variability	Variation in task response times	Improve response-time consistency	Assesses performance stability

An outcome of this work will be a better comprehension of which type of processing (edge vs. cloud) is more appropriate for different applications. This framework could potentially decrease the need to remotely communicate with a server to complete a task if that task could have been completed at the local level. Additionally, this framework may allow for the allocation of computationally expensive tasks to cloud-based systems when there are no constraints on bandwidth/latency and the task has sufficient time to complete. Although it appears as though there would be an increase in efficiency, improvements in efficiency are dependent upon several factors such as the type of workload being processed, the hardware available to process the workload, the current performance of the network, and the decision rules applied to determine where to process the workload. To obtain valid data regarding the effectiveness of this framework, experimental testing needs to occur across a wide variety of conditions including network congestion, insufficient resource allocation at the edge, and variable task arrival rates. Testing repeatedly can provide quantitative data concerning the consistency of the performance of the framework, as well as indicate when neither edge nor cloud processing can meet the required deadline. Data obtained through these tests can assist in refining decision criteria used to allocate resources.

X. CONCLUSION

The growing need for fast and dependable computing has provided a major hurdle for selecting the best location for processing when there are constraints on the time it takes to process. Both edge and cloud computing offer unique benefits; however, both have their own drawbacks. For instance, edge computing can provide low-latency processing, allow users to analyze data locally, and reduce the reliance on remote connectivity. On the other hand, cloud computing offers users access to large quantities of scalable computing power, centralizes user data storage, and provides users the ability to perform complex

analysis of data. In this paper we developed a simple decision model that would allow us to determine whether edge or cloud computing should be used based upon the needs of the application and the environment in which it operates. The factors considered include response-time requirements, network latency, available bandwidth, computational load, energy consumption, data privacy, and resource availability. This simple decision model provides a framework that allows us to place tasks in the most efficient manner throughout the various types of distributed computing environments.

The main value of the new framework being proposed is that it focuses on providing simple, flexible, and clear decision-making processes. Instead of using only very expensive computer-based optimization methods to determine which processing option is best for a given task, the framework uses measurements of how well the system is running (operating conditions) and pre-determined rules for making decisions about what processing options are available to help choose the right one. When a task needs an instant answer, the framework can assign the task to an edge device as long as there are enough local resources available to run it. On the other hand, if a task requires a lot of computing power and takes some time to finish, the framework can direct the task to a cloud-based server as long as the delay caused by sending the data across the network doesn't cause problems. Also, the framework recognizes that privacy concerns and resource constraints are important factors in choosing where to process tasks. In addition, the framework allows for regular reviews of the chosen processing option as workload, network conditions, or processing ability change. Therefore, this type of processing framework may be useful for many different types of applications including smart city systems, automated factories, connected health care systems, smart transportation systems, and real-time monitoring systems. However, as noted above, determining where to process a particular task depends on the specific characteristics of the task itself, and therefore, neither edge- nor cloud-based processing is necessarily better than the other in every case.

This study also identified performance measures needed to determine how well the framework works in practice. Performance measures such as average response time, percentile latency, deadline satisfaction, energy use, resource use, task completion rate, network load, and decision load give multiple ways to look at system performance. To fully evaluate the proposed framework, it should be compared to edge-only processing, cloud-only processing, and appropriate existing task offloading strategies. Experiments must include varying levels of workload intensity, network condition, device capability, and application deadlines. Comparing these areas will help show if simple decision-making rules help place tasks on devices better than simple decision-making rules while avoiding large amounts of additional monitoring and/or processing. Since the framework proposed in this paper is a concept, there is no proof that it is effective based upon experimental data. All statements about performance improvement, energy savings, or reliability advantages must be considered tentative until there is data showing the framework works as expected.

Future work will include exploring additional ways to make the framework more precise and flexible. Predictive models based on current operational metrics can be used to predict network delay and processing demands. Adaptive threshold settings may assist the system in adapting to workload changes. Security aware task placement can be extended to include the cost of encryption, authentication requirements, and privacy considerations in the process of making processing decisions. Additional areas of investigation can include workload partitioning, cooperative edge devices, and distributed processing on multiple cloud and edge platforms. The development of these capabilities must maintain the framework's lightweight characteristics and not introduce unnecessary complexity that would reduce the value of the decision making improvements. Additionally, experimental studies should investigate the overhead associated with switching between processing locations, measurement uncertainty, task dependencies, and fallback procedures if neither processing location meets the needs of the application. Deployment testing within realistic environments can provide more substantial evidence regarding the scalability, robustness, and applicability of the framework for various latency sensitive applications.

The developed lightweight decision framework is designed to provide users with a structured method to evaluate several key trade-offs in selecting whether to process data locally at the edge or remotely in the cloud. These trade-offs include; responsiveness (the speed at which data can be processed), computational capacity (the amount of data that can be processed at once), energy efficiency (the amount of power consumed), privacy (the protection of user's personal information), and resource availability (the amount of processing power available). The primary benefit of the proposed lightweight decision framework is that it organizes the relevant decision factors in an understandable manner that allows each user to adapt them to meet their specific needs based on the requirements of their application. By allowing users to make context-dependent decisions about where to run a particular application, the framework provides users with a useful baseline to improve the management of their

distributed computing resources. However, before the proposed lightweight decision framework can be widely adopted, several issues must be addressed.

XI. REFERENCES

[1] Bonomi, F., Milito, R., Zhu, J. and Addepalli, S. (2012) 'Fog computing and its role in the Internet of Things', Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16.

[2] Buyya, R. and Dastjerdi, A.V. (eds.) (2016) Internet of Things: Principles and Paradigms. Cambridge, MA: Morgan Kaufmann.

[3] Chiang, M. and Zhang, T. (2016) 'Fog and IoT: An overview of research opportunities', IEEE Internet of Things Journal, 3(6), pp. 854–864.

[4] Deng, R., Lu, R., Lai, C., Luan, T.H. and Liang, H. (2016) 'Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption', IEEE Internet of Things Journal, 3(6), pp. 1171–1181.

[5] Dinh, T.Q., Tang, J., La, Q.D. and Quek, T.Q.S. (2017) 'Offloading in mobile edge computing: Task allocation and computational frequency scaling', IEEE Transactions on Communications, 65(8), pp. 3571–3584.

[6] Fadlullah, Z.M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T. and Mizutani, K. (2017) 'State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems', IEEE Communications Surveys & Tutorials, 19(4), pp. 2432–2455.

[7] Fernando, N., Loke, S.W. and Rahayu, W. (2013) 'Mobile cloud computing: A survey', Future Generation Computer Systems, 29(1), pp. 84–106.

[8] Gai, K., Qiu, M., Zhao, H., Tao, L. and Zong, Z. (2016) 'Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing', Journal of Network and Computer Applications, 59, pp. 46–54.

[9] García-Valls, M., Cucinotta, T. and Lu, C. (2014) 'Challenges in real-time virtualization and predictable cloud computing', Journal of Systems Architecture, 60(9), pp. 726–740.

[10] Goyal, S. and Carter, J. (2004) 'A lightweight secure cyber foraging infrastructure for resource-constrained devices', Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications, pp. 186–195.

[11] Satyanarayanan, M., Bahl, P., Caceres, R. and Davies, N. (2009) 'The case for VM-based cloudlets in mobile computing', IEEE Pervasive Computing, 8(4), pp. 14–23.

[12] Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016) 'Edge computing: Vision and challenges', IEEE Internet of Things Journal, 3(5), pp. 637–646.

[13] Mao, Y., You, C., Zhang, J., Huang, K. and Letaief, K.B. (2017) 'A survey on mobile edge computing: The communication perspective', IEEE Communications Surveys & Tutorials, 19(4), pp. 2322–2358.

[14] Mach, P. and Bečvář, Z. (2017) 'Mobile edge computing: A survey on architecture and computation offloading', IEEE Communications Surveys & Tutorials, 19(3), pp. 1628–1656.

[15] Abbas, N., Zhang, Y., Taherkordi, A. and Skeie, T. (2018) 'Mobile edge computing: A survey', IEEE Internet of Things Journal, 5(1), pp. 450–465.

[16] Liu, J., Mao, Y., Zhang, J. and Letaief, K.B. (2016) 'Delay-optimal computation task scheduling for mobile-edge computing systems', 2016 IEEE International Symposium on Information Theory (ISIT), pp. 1451–1455.

[17] Sardellitti, S., Scutari, G. and Barbarossa, S. (2015) 'Joint optimization of radio and computational resources for multicell mobile-edge computing', IEEE Transactions on Signal and Information Processing over Networks, 1(2), pp. 89–103.

[18] Chen, X., Jiao, L., Li, W. and Fu, X. (2016) 'Efficient multi-user computation offloading for mobile-edge cloud computing', IEEE/ACM Transactions on Networking, 24(5), pp. 2795–2808.

[19] Kosta, S., Aucinas, A., Hui, P., Mortier, R. and Zhang, X. (2012) 'ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading', Proceedings of IEEE INFOCOM, pp. 945–953.

[20] Chun, B.-G. and Maniatis, P. (2009) 'Augmented smartphone applications through clone cloud execution', Proceedings of the 12th Workshop on Hot Topics in Operating Systems (HotOS XII).

[21] Cuervo, E., Balasubramanian, A., Cho, D.-K., Wolman, A., Saroiu, S., Chandra, R. and Bahl, P. (2010) 'MAUI: Making smartphones last longer with code offload', Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62.

[22] Fernando, N., Loke, S.W. and Rahayu, W. (2013) 'Mobile cloud computing: A survey', Future Generation Computer Systems, 29(1), pp. 84–106.

[23] Kumar, K., Liu, J., Lu, Y.-H. And Baraga, B. (2013) 'A survey of computation offloading for mobile systems', Mobile Networks and Applications, 18(1), pp. 129–140.

[24] Flores, H. and Srirama, S.N. (2013) 'Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning', Proceedings of the 4th ACM Workshop on Mobile Cloud Computing and Services, pp. 9–16.

[25] Satyanarayanan, M. (2017) 'The emergence of edge computing', Computer, 50(1), pp. 30–39.

[26] Varghese, B. and Buyya, R. (2018) 'Next generation cloud computing: New trends and research directions', Future Generation Computer Systems, 79, pp. 849–861.

[27] Yi, S., Li, C. and Li, Q. (2015) 'A survey of fog computing: Concepts, applications and issues', Proceedings of the 2015 Workshop on Mobile Big Data, pp. 37–42.

[28] Roman, R., Lopez, J. and Mambo, M. (2018) 'Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges', Future Generation Computer Systems, 78, pp. 680–698.
[29] Balaji, R. (2025) 'Autonomous SRE: SLO-Aware Multi-Agent Root-Cause Analysis and Verified Self-Healing for Cloud-Native Systems', ESP Journal of Engineering & Technology Advancements, 5(4), pp. 211-221. doi: 10.56472/25832646/JETA-V5I4P128.
[30] Shi, W. and Dustcart, S. (2016) 'The promise of edge computing', Computer, 49(5), pp. 78–81.
[31] Dong, S., Tang, J., Abbas, K., Hour, R., Kamruzzaman, J., Rutkowski, L. and Buyya, R. (2025) 'Task offloading strategies for mobile edge computing: A survey', Computer Networks, 255, article 110791.