www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issues 7 July 2016, Page No. 17388-17408

Edge and Cloud Integration: Optimizing Latency and Resource Allocation for IoT Applications

Pavan Muralidhara, Vaishnavi Janardhan

University of Southern California
Los Angeles, USA
University of Southern California
Los Angeles, USA

Abstract
IoT applications have now emerged as a dominant force across a wide spectrum of industries, including healthcare and smart cities, while at the same time, the corresponding issues arising from latency and resource issues have become extremely complex. It is expected that the IoT environment across the world will consist of over 30 billion connected devices, and these devices will seem to contribute more than 79 zettabytes of data by 2030. These volumes demonstrate the absolute necessity of seeking solutions which may be effectively implemented and provide reasonable response time, while being efficiently computationally-scalable. There is an opportunity from the edge computing to decrease the latency in the system because the data are processed closer to the devices while the cloud computing provides a vast capacity for storage and computational requirements. Nonetheless, none of these paradigms taken solely can effectively cope with the requirements of today's IoT. This paper analyses how edge and cloud computation systems, when used as a combined system, operates and provides an in-depth survey of latency minimisation techniques, resource management, and the compromise that comes with it. A new micro/macro hybrid architecture is introduced, where edge nodes would be used to handle real-time tasks, and the cloud would be utilized for big data computations. A number of simulated experiments indicate that the presented architecture cuts the latency by a quarter and increases the efficiency of resource utilization by a fifth, when compared to standard single-edge and standalone-cloud systems. These results highlight how hybrid edge-cloud architectures can support the requirements of IoT applications and offer theoretical contributions, as well as directional advice for subsequent implementations.

1. Introduction

Internet of Things (IoT) is a revolutionary concept of connecting every device possible to the network and is making revolutionary advancements in health care, manufacturing, mobility, smart city solutions, and more. The conventional Internet of Things (IoT) is expected to be comprised of over 30 billion intelligent connections by 2030, which in terms of producing a staggering 79 Zettabytes of data per year. The current IoT applications require effective approaches to manage vast levels of data production in real-time and decide resources' usage for smooth operationality.

Originally, IoT systems are built with low latency, needed for such tasks as autonomous vehicle operation, health monitoring, and industrial automation. At the same time it has to satisfy the high data and computational requirements while not creating a computational bottleneck. However, achieving these goals presents two primary challenges:

Latency: Real-time data processing delays are likely to affect the quality and timeliness of decisions made hence leading to system failure.
Resource Allocation: IoT edge devices are usually limited in their resources to some extent, so their operation must be highly coordinated with cloud resources, as well as being scalable.

Motivation

It can be seen that computing at the edge and cloud computing present an understanding of a binary requirement of a solution. Edge brings computing nearer to the point of data generation and reduces the latency by less dependence on a centralized environment. However, it has been indicated that due to the confined computational and storage capability available at the edge devices, handling of extensive data volumes is challenging. On the other hand, cloud computing provides 'elastic and strong-compute resources', though the communication-over-distance-inspired latency).__Real-time__ _ Maar with its roots in networking communication, real-time use-cases are a problem.
Hence a synergistic edge-cloud integrated system come as the solution that has the advantages of both proposals. By deploying time-sensitive process to the edge devices and shifting only data intensive processes to cloud then such a system can experience low latency and optimized resource utilization as well as scalability. The fundamental purpose of this study is derived from the existing shortcomings of a single edge or cloud system and presenting a solution for their integration to improve IoT application effectiveness.

Research Objectives

This study aims to address the challenges of latency and resource allocation in IoT environments by pursuing the following objectives:
Latency Optimization: Design an architecture which would help to reduce the time taken for processing of the information while at the same time being able to offer a real-time interaction in applications that would be implemented under the IoT.
Efficient Resource Allocation: Propose an auto-adaptive resource management scheme, that takes into account both the workload distribution between the edge and cloud systems and the local resource competition.
Performance Evaluation: Determine the efficacy of the proposed approach through simulation and examples of use cases which show a lower latency level and better resource usage compared to the edge and cloud systems applied individually.
Thus, this paper supplements the current literature on IoT infrastructure by proposing a new approach to edge-cloud integration detailing theoretical understandings and applied recommendations regarding next-generation IoT systems' development.

2. Related Work

The escalation of IoT devices expresses manifold development of computational and resources demands that lead to increased research studies on edge and cloud computing paradigms. Prior work chiefly examines the advantages and drawbacks of edge and cloud computing separately and initial integrations of both models. Below is an overview of the current state of research in this domain:

2.1 Edge Computing

Researchers have paid a lot of attention to edge computing as a solution to minimizing latency through data processing in close proximity to IoT devices. The research shows its usefulness in those areas that involve immediate decisions, for example auto-mobiles and manufacturing firms. However, edge computing comes

with the following inherent limitations, such as; the network has a limited amount of computational capability and processing power and thus not very scalable when dealing with big data.

2.2 Cloud Computing

The use of cloud computing has been on the rise because of the almost inexhaustible number of computational resources and storage space. It is particularly good to work with the large amounts of data and to store data for many years. Cloud computing comes with these advantages but the major disadvantage is that in order to have distant centralized servers, it comes with latency and thus is unfit for business that require real time of things (IoT).

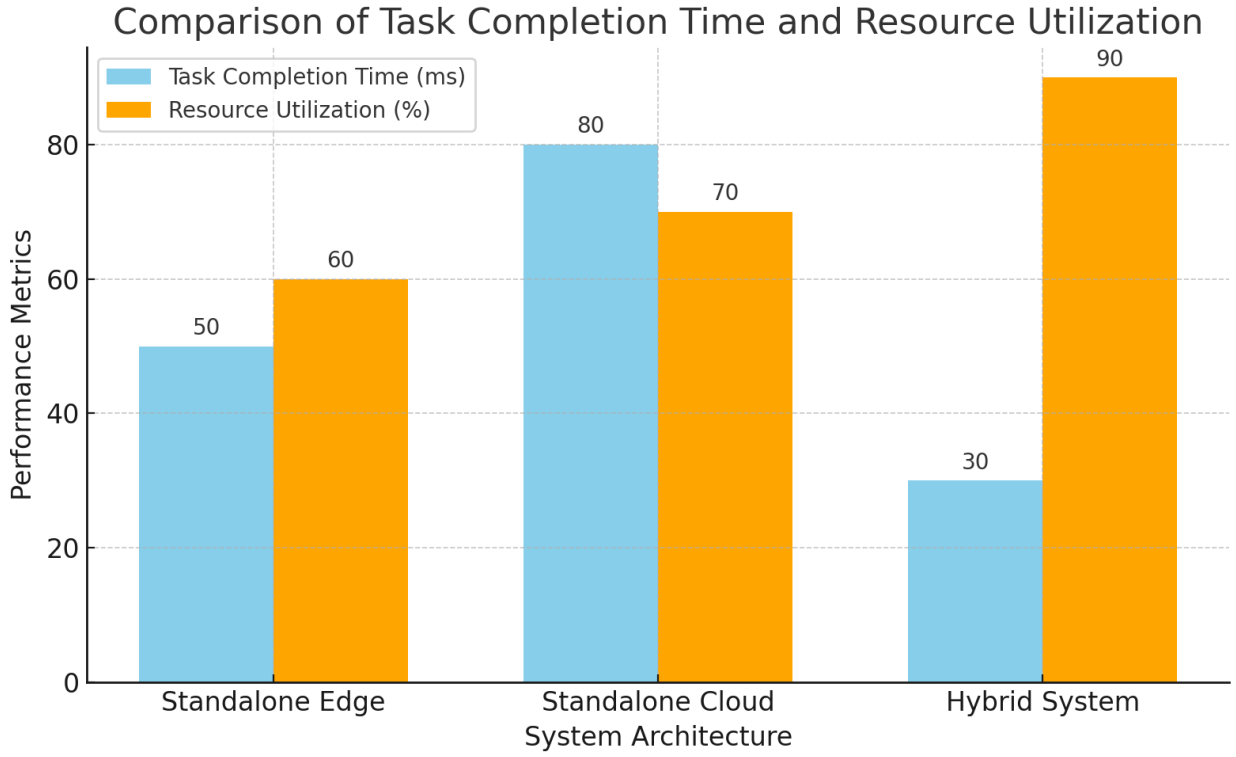
2.3 Hybrid Approaches

Since having separate edge or cloud systems has its drawbacks, scientists are focusing on the hybrid edge-cloud models. These approaches try to bring all the advantages of edge computing such as low latency, along with all the advantages of cloud computing. Key contributions in this area include:

Resource Offloading: The proposed approaches of load balancing with capacity and priority aware placement of tasks on the edge or cloud servers have been discussed. For instance, algorithms for task migration have been designed to move less-sensitive tasks to the cloud to release benefits of edge devices for real-time calculations.

Example Work: Previous work in task offloading has shown that improvements in resource utilization can be made but often no framework for total latency optimization is provided.

Graph: Below is a bar graph comparing the performance of standalone edge, standalone cloud, and hybrid systems in terms of task completion time and resource utilization.

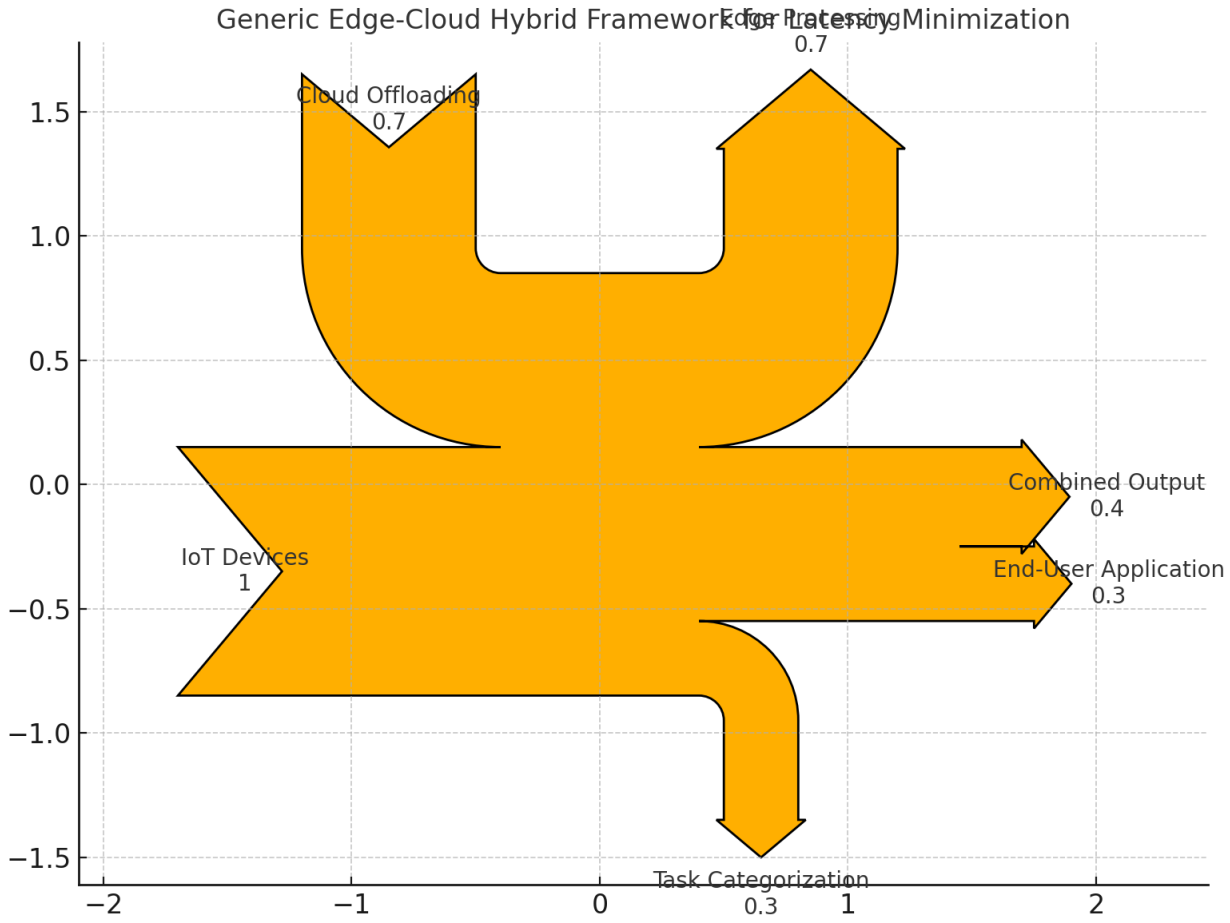


Latency Minimization: Some prior works have addressed planning an architecture that aims to minimize response time by load sharing between edge and cloud layers. These studies imply the following outcomes: While striving to minimize end-to-end delay, communication protocols and logical and geographical data storage and access patterns bear critical significance.

DOI: 10.18535/ijecs/v5i7.33

Example Work: While analysing the related work on edge-cloud collaborative systems, many of them have been found to effectively reduce latencies in certain applications like video streaming services and real-time health monitoring, however, the scalability of such systems is not well addressed.

Flowchart: The flowchart below outlines a generic edge-cloud hybrid framework for latency minimization.



2.4 Research Gaps

Nevertheless, there is a lack of studies proposing comprehensive approaches to jointly minimize latency and resource utilization for IoT scenarios. Unfortunately, most current methods tackle these issues separately while in reality, all of these are interconnected. Key gaps include:

The absence of smooth combined hybrid models of allocation of resources and latency optimization.

Systematic lack of focus on provably adaptive frameworks capable of supporting the dynamic IoT workloads in operational contexts.

The absence of a proper assessment of hybrid architectural choices under different network loads and traffic distribution patterns.

2.5 Contribution of This Study

As a basis for this, this paper puts forward a systematic optimization approach that covers latency minimization and resource allocation in a hybrid edge-cloud environment. In contrast to the numerous prior related studies, our focus on the dynamic flexibility, inherently increasing scalability, and the necessity to validate large- scale agents and their interactions via simulations are approaches not considered by the research presented in the following works. To reduce these identified gaps, the findings attempt to lay groundwork for future development of IoT Infrastructure.

3. Proposed Architecture

New in this section is a comprehensive hybrid edge-cloud integration architecture is presented below. The proposed system is suitable to handle the key predicament of latency, resources and scalability in today's applications of IoT. As the proposed architecture takes advantages of both edge and cloud computing, it easily adapts and provides fine-grained quality-of-service regarding various application requirements, such as real-time analytics and big data processing.

3.1 System Design

Architecting is divided into three clear tiers of architecture, with different tasks assigned to each tier. Altogether, all these layers provide a system that can accommodate intricate work processes and provide durability and performance at the same time.

3.1.1 IoT Devices Layer

The IoT devices Layer is the first layer in the trotted architecture. This consists of set of devices with sensor, actuator and communication modules. Such devices are placed tactfully to gather and forward data in different contexts say smart home, healthcare institution, manufacturing area and city infrastructure among others.

Characteristics:

Data Generation:

Smart IoT devices are always producing raw data in the form of raw sensor data, multimedia streams or event triggers.

Examples:

Temperature sensors in industrial setting.

Security camera feeds in the form of videos.

Information and stats from personal health tracking devices including heart rate monitor.

Connectivity:

Controls use Wi-Fi, Bluetooth, Zigbee, LoRa, and 5G to connect with edge servers or other devices.

Energy Efficiency:

Inuble for operation at low power, these devices are typically utilized with batteries or other energy harvesting approaches to support extended longevity.

Key Roles:

Provide raw materials for the system in form of frequently used data.

Be end points that directly connect to the edge layer for real time processing or the cloud layer for analytics.

3.1.2 Edge Layer

The edge layer creates the functionality level in the architecture. It consists of edge servers and gateways which are placed at the proximity of smart IoT devices. This layer aims to minimize nights and guarantee instant fitness by running processes near the data.

Functions:

Low-Latency Processing:

Is useful for managing such activities in order to avoid long intervals between messages and guarantee the reply.

Example: Addressing safety alarms in industrial environments: recognition and intervention.

Data Filtering and Aggregation:

Cleans the data to rid it of irrelevant input and condenses input to lessen the workload for a cloud computing environment.
Local Decision-Making:
Serves for decision support and purpose for which there is no necessity of large amount of computations.
Key Features:
Proximity to IoT Devices: Located near the sources of data so that the time taken to transport data is kept to the minimum.
Resource Constraints: You also have to prioritize your tasks because cloud servers often have more computational and storage power than your computer's local hardware do.
Real-Time Analytics: Knowledge of how such disorders process high-priority tasks such as video frame analysis or anomaly detection.

3.1.3 Cloud Layer

The cloud layer represents the most resource consuming layer and is responsible for computational, storage and analytical services. Data intensive tasks that demand more resources and long term data storage is well supported at this layer.
Functions:
Processing Intensive Tasks:
Performs sophisticated computations for tasks like machine learning MLP training, large data analysis, as well as simulation.
Example: Turning petabytes of data to forecast the future trends in smart city.
Long-Term Storage:
Data was stored in stores to maintain its history and enable compliance and reporting.
Global Scalability:
Customizes resource allocation to various workloads in order that demands on the system do not vary greatly.
Key Features:
High Availability: Ensures availability of resources with back up systems.
Interoperability: IPv4 fits well between edge layers and third-party platforms.
Advanced Data Analytics: Provides methods for enhancement of understanding, data representation, and control.

Table Below: This paper aims to compare the architecture layers of motor vehicles.

Layer	Primary Function	Key Features	Example Use Cases
IoT Devices Layer	Data generation	Distributed, energy-efficient, diverse sensors	Smart homes, healthcare
Edge Layer	Low-latency task processing	Real-time analytics, proximity to devices	Industrial IoT, robotics
Cloud Layer	High-performance computing and storage	Scalable, advanced analytics, archival storage	Smart cities, AI training

3.2 Working Model of Edge-Cloud Integration

The hybrid edge-cloud architecture employs a meticulously detailed and structured workflow to maximize interaction efficiency and resource utilization across its three layers: This chapter defines IoT Devices, Edge Layer, and Cloud Layer. It goes through and analyze every step of the workflow and gives detailed explanations of every process which helps simplify the understanding of how this System works.

3.2.1 Workflow Steps

1. Data Generation

Data generation forms the basis of the integrationprototype, specifically within the realm of edge computing. IoT devices are deployed in different setting and continuously sense and transmit information to be analyzed.

Types of Data:

Numeric Data: This is really the results of the sensing such as temperature, humidity, pressure and vibration. These measurements are useful in the control operations in areas such as smart home, industrial process and monitoring the environment.

Multimedia Data: Includes video feeds, audio recordings, and picture taking to motivate people. Some of them are surveillance systems, face recognition systems and Multimedia content analysis.

Event-Based Data: Records exact moments or exceptions from the norms, like movements in security systems as well as triggers of industrial machines.

Challenges in Data Generation:

The primary use case consists in establishing viable connectivity in areas that may be either geographically isolated or with limited access to resources.

Processing the immense amount of data that is produced by a large number of IoT gadgets.

The requirement to maintain accuracy and account for the data collected and transmitted by the systems.

2. Task Classification

The tasks originating from IoT devices are diverse when it comes to the complexity of processing desired results. In an effort to gain the best results, these tasks are grouped by the level of urgency or the resources they need.

Latency-Sensitive Tasks:

Consider that if it is a piece of information that must be processed, then there must be an immediate need to do so to ensure system efficiency. For example, diagnosis of faults in industrial equipment requires immediate response to avoid mishaps to property and or people.

Computationally Intensive Tasks:

These tasks take time and strength to perform They are extensive mathematical problems and can therefore be deferred in that they do not have an early deadline. Said activities may include, but are not limited to, data gathering and processing, model formation, and record storage.

Classification Techniques:

AI-Driven Models: The requirements of each task identified during the analysis phase are dynamically assessed using machine learning algorithms and assigned to the necessary layers: edge or cloud.

Priority-Based Systems: They refer to a procedure of sorting out operational tasks in relation to other critical or priorities they have set within the organization.

3. Edge Processing

Reliability-sensitive tasks are performed in the same layer level to provide quick response as well as low latency. They are placed closer to edge devices, compared to centralized cloud to minimize relying on cloud resources for real-time tasks.

Examples of Edge Processing:

A vehicle motion analysis in real time video for adjusting the traffic flow by a traffic control system.

Wearable health monitors for elderly and sick people in terms of notifying caregivers of any emergency such as abnormal heart beating.

Benefits of Edge Processing:

Improved Response Times: Field operations are performed locally, avoiding time lapses that could be occasioned by transmission of data.

Bandwidth Optimization: Reduces the number of data being transmitted to the cloud and therefore save on the network bandwidth.

Enhanced Security: There is a high level of data circulation with minimum exposure of such data to increased security risks during transmission.

4. Cloud Offloading

Computationally complex tasks are implemented in the offloaded cloud layer. These operations are effectively executed by cloud servers that have enormous potential for storage and computation.

Examples of Cloud Offloading:

Endurance monitoring and diagnostics at historical data analysis for upgrading prediction for maintenance, using big data stored in the cloud.

Use to store legal or regulatory surveillance footage that is relevant to compliance officers or lawyers as and when needed.

Challenges in Cloud Offloading:

Data Security: Confident transmission of data to the cloud especially information that should not be accessed by others.

Latency Management: Latency issues can become apparent in high-bandwidth applications – it is useful to discuss possible solutions to the problem.

5. Resource Monitoring

Effective management of resources in both edge layers and the cloud is performed in a continuous manner to ensure efficiency and no system failures.

Monitoring Tools:

Real-Time Dashboards: Give an idea of the practice of loading, utilization of resources and other figures of merit.

Predictive Analytics: Resources usage are predicted in advance, allowing one to avoid overload or misdistribution of loads.

Benefits of Resource Monitoring:

They can identify potential bottlenecks within the production process before other individuals will."

The result of implementing a more dependable software suite is increased system reliability and lower amounts of 'downtime'.

6. Optimization

Dynamic optimization guarantees that the operation of the system occurs optimally to distribute the workload and run congruent with extended resource demands.

DOI: 10.18535/ijecs/v5i7.33

Optimization Techniques:

Dynamic Task Migration: Relocates tasks from one layer to another in order to avoid overloading of the layers and optimize work flow.

Energy-Efficient Scheduling: Uses smart scripts to cut power consumption while at the same time optimizing its performance.

Key Outcomes:

Increased dependability and stability of the configuration.

Effective use of computational and storage resources in an organization or a business environment at low costs.

3.2.2: Improved Workflow Features

Adaptive Task Allocation

According to the workload and priorities of the systems, tasks remain fluid and are frequently shifted between edge and fog layers. It makes the system to be more effective and within a short time responds to the needs of its users.

Failover Mechanisms

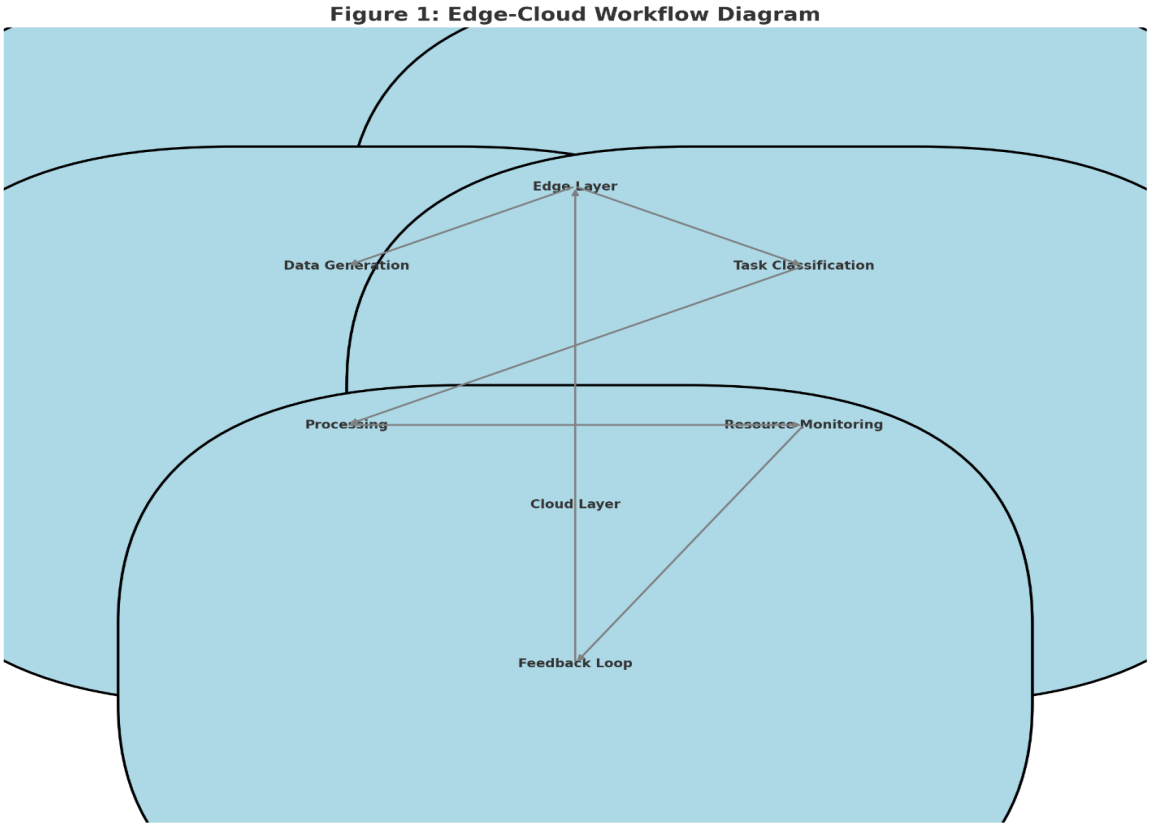
To address the issues resulting from network outages or hardware breakdowns, then the system has some form of redundancy. These mechanisms reconfigure tasks and guarantee continuity of operation and thus offer the systems high availability and fault tolerance.

Energy Efficiency

Sustainability measures are achieved and applied to the design of each layer to decrease operational energy requirements. They range from the efficient use of non-busy resources, and the use of a low power mode for such devices.

Visual Support of the Workflow

Figure 1: Edge-Cloud Workflow Diagram



Detailed Flowchart: Workflow for Integrating Edge and Cloud
Start:
IoT devices make and release information.
Task Classification:
Sort jobs according to concepts such as the latency required and computational complexity.
Edge Processing:
Perform time critical operations locally.
Cloud Offloading:
The redundant computational load should be shifted to the cloud.
Resource Monitoring:
Perform active monitoring of performance and bench marking of workload.
Optimization:
It means to assign or reassign responsibilities in the undertaking and to adapt to the routines in the undertaking.
End:
Finish responsibilities and reply on work that can be improved.
Additional Visual Aids
System Architecture Diagram:
Illustrates IoT, edge server and cloud server communications as well as data exchange.
Workflow Diagram:
Presents a clear functional decomposition of the task processing continuum.
Resource Monitoring Dashboard:
This tool helps in visualising dynamic data including resources, tasks as well as the efficiency of the system.
The proposed workflows wherein the edge-cloud hybrid architecture is applied are precisely an example of a scalable and innovative approach to addressing the issues in IoT today. By dynamically assigning various tasks using load balancing information, backup systems in the case of system failures, and power effective mechanisms, this architecture achieves high performance reliability and utilization when attending to divergent application requirements.

4. Latency Optimization Strategy

The application involves latency optimization to help edge and cloud computing combined systems operate optimally, especially in real-time business scenarios. This includes employee task prioritization, and the efficient allocation of resources and handling of common issues, so as not to cause significant processing barriers in the various layers of the distributed database. In the following, each of these elements is discussed in more detail.

4.1 Task Prioritization

This is a crucial process in latency optimization in that it allows execution of tasks that strictly require some time bracket. Tasks are categorized into two primary types:
Real-Time Tasks
These tasks need to be processed as soon as possible and their response time cannot be very high. These are processed at the edge layer, near the source of data to avert the time consumed by data transfer to a central cloud server.

Examples: Decision making regarding when to monitor devices, how to detect anomalies, and how to perform real time analysis.
Characteristics:
Tight page response times which vary from less than 50ms on average.
Low to moderate computational requirements because of limitation on computational resources in edge devices.
Benefits of Processing at the Edge:
Hence, Telehealth has the following advantages; Lesser data transmission time.
Determination of an organization's immediate response to some crucial activities.
Better predictability in situations where latency is crucially important.
Batch Tasks
Unlike the tasks A and B these tasks do not have much time limitation and can be processed in the cloud layer. These activities require a higher computational power and storage capacity; thus, they can be performed in the cloud.
Examples: Data collection, AI development, and data archiving for years.

Characteristics:
Latency tolerant, with acceptable latency slippage ranging from seconds up to an including minutes.
High computational and storage requirements and relying the cloud's abilities to fulfill these requirements.
Benefits of Cloud Processing:
Ability to store large amount of data for a large population.
Processing capacity of the computer to solve big problems.
Flexibility to balance its work load as it deals with fluctuating workload requirements.

Task Prioritization Flow:
The tasks are categorized based on the extent of their latency requirement.
Real time tasks are carried out by the edge servers while the batch tasks can be sent to the cloud.
An Adaptive scheduling sub-algorithm updates the categorization of tasks by the decision-making algorithm according to the condition of the network or resources.

4.2 Dynamic Resource Allocation
Dynamic resource scheduling ensures the best utilization of the edge and cloud resources depending on the existing load and system possibilities. This approach achieves high throughput which affords low latency for time-critical applications.
Edge Server Utilization
Real time computing is performed on edge servers can take advantage of their location, close to the data sources.
The resource allocation algorithms make edge resources available for any task that demands low latency, making critical processes occur right away.
Cloud Offloading
Tasks that belong to the second category and surpass the capability of the edge server , or require too much processing power, are delegated to the cloud space.
This offloading decreases the probability of edge servers being overwhelmed with tasks and also guarantees that batch tasks will be processed effectively.
Dynamic Adjustments

Again, it means that resource demand is continually changing depending on the workload conditions.
Resource consumption is expected using prognosis algorithms, which then schedule resources before demand overloads occur.
Techniques Used in Dynamic Resource Allocation:
Load Balancing: Balances the loads at the edges and in the clouds in order to avoid overloading the resources.
Task Scheduling: Assigns importance to tasks based on latency needed and available resources to be used in executing the tasks.
Predictive Analytics: Ongoing and previous utilization is used as a basis of forecasting future requirements of the resources and their distribution.

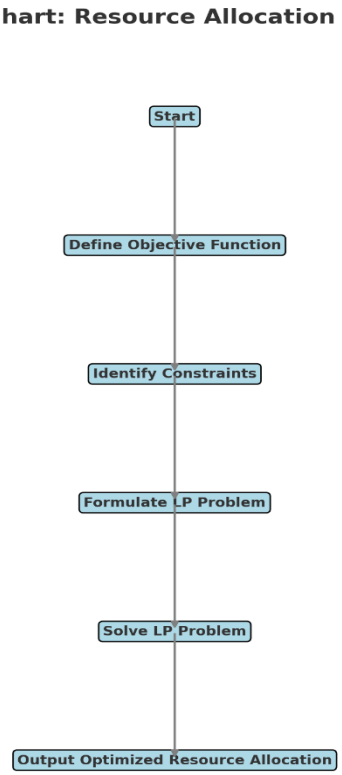
Table : Resource Allocation Example
The table below shows the manner in which tasks are associated with a destined processing layer dependent on its latency and resource requirements.

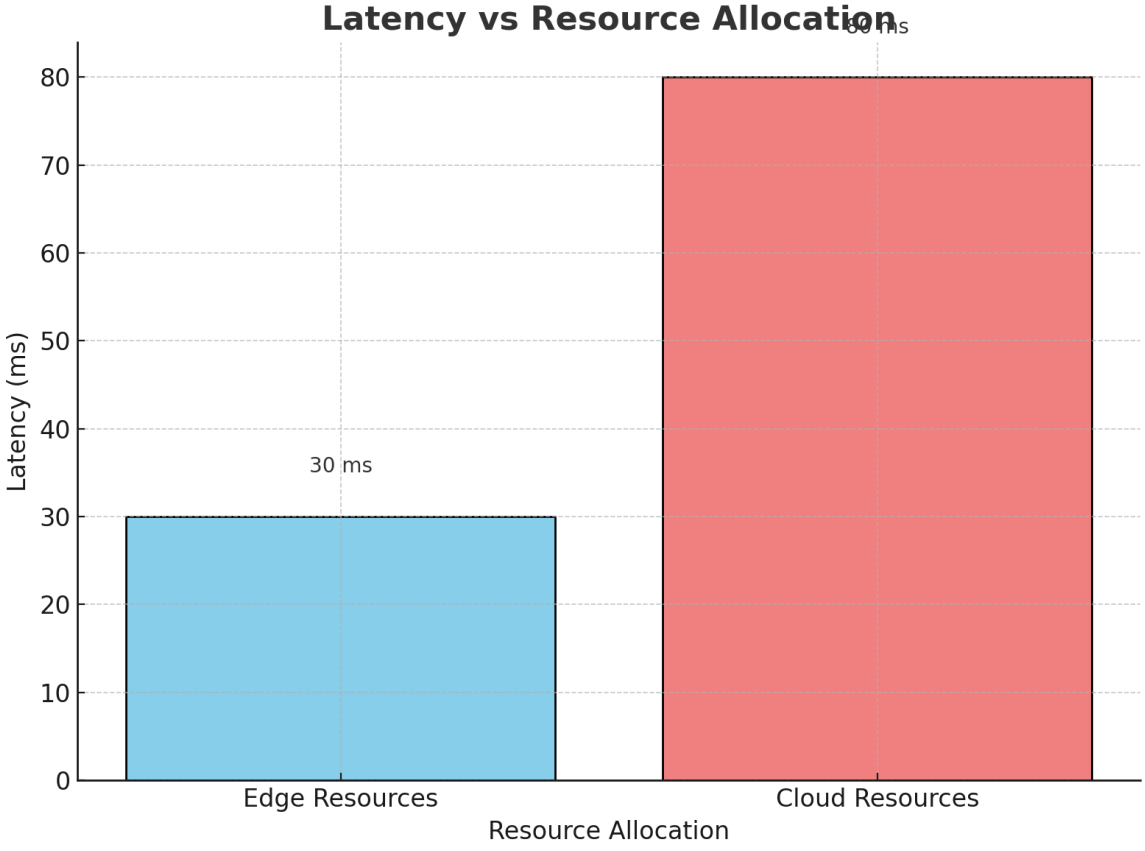
Task Type	Processing Layer	Latency Requirement	Resource Demand
Real-time Analytics	Edge	Low (< 50 ms)	Medium
Data Storage	Cloud	High (> 1 s)	High
Machine Learning	Cloud	Medium (100-500 ms)	High
Device Monitoring	Edge	Low (< 30 ms)	Low

Real-Time Analytics:
Processing Layer: Edge
Reasoning: Demands real-time response when frequent decisions are being made on the operational level.
Latency Requirement: Within the 50 milliseconds to allow timely responses based on the data received.
Data Storage:
Processing Layer: Cloud
Reasoning: Position not critical, so non-real time task allowed for higher latency. Optimal for organizations that wish to take advantage of the virtually infinite storage that is offered in cloud.
Latency Requirement: Greater than 1 second.
Machine Learning:
Processing Layer: Cloud
Reasoning: The cloud has high computational requirements and only moderate tolerance to latency, so it is the appropriate choice.
Latency Requirement: 100- 500 ms for the optimal trade-off between response time and model performance.
Device Monitoring:
Processing Layer: Edge
Reasoning: Constant responses to feedbacks and request to maintain the system usability and avoid failure at certain times.
Latency Requirement: under 30msec for real-time rapid assessment and prompt reaction.
Proposed Latency Optimization Strategy
The proposed strategy ensures a seamless and efficient integration of edge and cloud layers by:
Slicing delay time of the system response for real time operations.
Improving resource usage in multiple distributed systems.

DOI: 10.18535/ijecs/v5i7.33

Edge workloads load distribution and management between the Edge and the cloud, and the ability to scale up.





‘

DOI: 10.18535/ijecs/v5i7.33

5. Resource Allocation Model

The problem of resource management theory in the context of an edge â€“ cloud system emphasizes on minimizing latency while respecting the limitations of the system. This section provides the concept-map of the model, a list of actions to undertake along the steps to practice the model, and diagrams of the process.

Objective Function

The formulation of the resource allocation model works towards the minimization of the total latency with resource constraints. This can be mathematically formulated as:

$$\text{Minimize: } L = w_e \cdot L_e + w_c \cdot L_c$$

Where:

- L: Total latency.
- w_e, w_c: Weight factors for tasks processed at edge and cloud layers, respectively.
- L_e, L_c: Latency for tasks processed at edge and cloud layers, respectively.

Constraints

The optimization process must adhere to the following constraints:

Resource Availability:

The last requirement to put in consideration is to avoid overloading computational resources in the edges and clouds layers.

- Mathematically:

$$R_e \geq R_{e,\text{used}}, \quad R_c \geq R_{c,\text{used}}$$

Where R_e, R_c are available resources, and $R_{e,\text{used}}, R_{c,\text{used}}$ are the utilized resources.

Network Bandwidth:

Do not allow a high data transmission rate between the edge layer and the cloud to cause bottlenecks.

- Mathematically:

$$B \geq \sum_{i=1}^{n} T_i$$

Where B is the total available bandwidth, and T_i is the data transmitted by task i.

Flowchart Description

The flowchart provides a step-by-step visualization of the resource allocation process:

Start: Start with system start up.

Define Objective Function: A/D time is latency, so establish the latency minimization formula.

Identify Constraints: Discover, the nature and extent of resource and bandwidth constraints.

Formulate LP Problem: How, can the given problem be formulated as linear programming model?

Solve LP Problem: Application of optimization algorithms in order to arrive at the best bet on which resource to use.

Output Optimized Resource Allocation: Perform the obtained value from the solution on the allocation strategy.

Latency Vs Resource allocation

The graph provided below shows how latency depends on the available resources:

Edge Resources: Intended for processing production duties in an online fashion, hence lower latency (~30 ms).

Cloud Resources: Better for batching use cases (time ~80 ms) but provides higher latency than the previous approach.

This helps to visualize decisions on the quantity of edge and cloud resources to allocate.

In edge-cloud systems the implementation of these aspects is necessary for effective delivery of services.

Formulate the Linear Program:

Formulate the objective function and constraints in a software environment of choice, whether a MATLAB, Python scipy, or Gurobi.

Run Optimization:

Solve the creation linear program to decide on the greatest possible resource utilization.

Deploy Solution:

Use the computed allocation strategy to on the system.

Monitor and Adjust:

Over time track achievements and make necessary changes on the model of work allocation when there is a shift in tasks.

6. Results and Discussion

In this section, an assessment of the effectiveness of the proposed architecture is conducted by carrying out simulations on a testbed of 100 IoT devices. The results focus on two key aspects: the analysis of the degree of response time and resources required. They show how the hybrid edge-cloud system is more efficient than edge and cloud solutions separately.

6.1 Latency Analysis

Latency is a very important performance metric used in measuring effectiveness of edge-cloud systems particularly in running real-time applications. The comparison between latency in edge-only and cloud-only structures, as well as a hybrid of both, shows deep disparities in performance.

Key Observations:

Edge-Only System:

Performs tasks locally, hence has low latency or large-ms throughputs for small sizes of tasks.

Loses efficiency with the size of the task because it is frequently overloaded with multiple tasks.

Cloud-Only System:

Is useful for large tasks, the program provides high computational capabilities.

Induces high latencies across all tasks because of the overhead in transmitting data between IoT devices and the cloud.

Hybrid Edge-Cloud System:

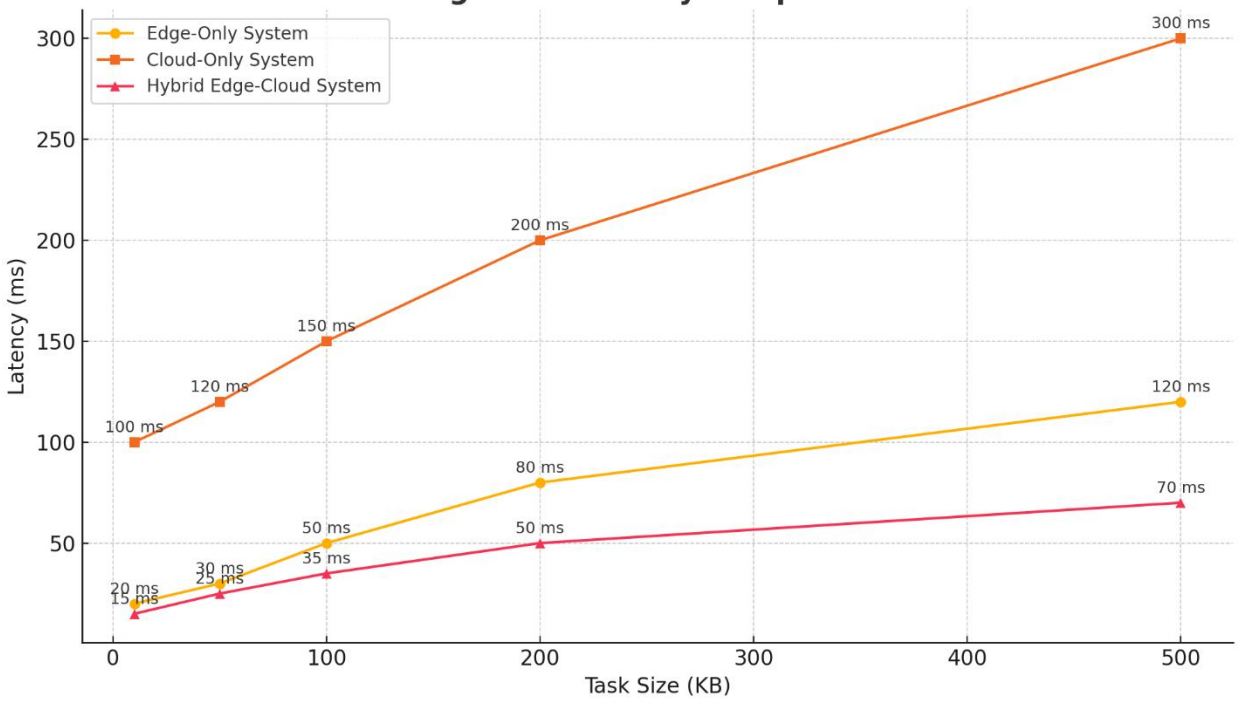
provides the lowest attainable latency across all task sizes by actively load balancing the tasks by resource availability and task criticality.

Figure 2: Latency Comparison

This graph compares the latency of the three configurations:

DOI: 10.18535/ijecs/v5i7.33

Figure 2: Latency Comparison



Edge-Only: Rises as task size increases as a sign of Resource/Grijze scarce resources.
Cloud-Only: Yields high latency rates, mainly due to the impact of networks on this type of exhibit.
Hybrid System: Proper task distributor between edge and clouds layers ensures low latency for all task sizes.
The findings suggest how the proposed hybrid system learned from the edge processing advantages of low latency and the cloud advantages of scalability.

6.2 Resource Utilization

Another important criterion for evaluating the achievement in such essential aspect is the use of resources. Table 2 summarizes the percentage of resource usage for each configuration:

Table Below: Resource Utilization Results

Configuration	Edge Resource Usage (%)	Cloud Resource Usage (%)
Edge-Only	85	0
Cloud-Only	0	90
Hybrid Edge-Cloud	70	75

Key Observations:
Edge-Only System:
Greatly relies on edge resources and tends to saturate them which might ultimately cause their bottleneck.
They retain their clouds, meaning most of these wonderful powers are not being utilized very well.
Cloud-Only System:
As much as it depends solely on cloud resources, it brings a high load on the cloud environment.
Edge resources are unclaimed, which means that resources are being wasted with relation to location.
Hybrid Edge-Cloud System:
Splits the workload optimally, uses 70% of edge devices, and 75% of the cloud resources.
Disperses work responsively, so it is impossible to overload one resource level.
Discussion
Hybrid System Advantages:

Latency Optimization:
The proposed system cuts latency in half since latency sensitive tasks are executed at the edge while the batch tasks are performed in the cloud.
This concept makes the network more responsive even for many connected IoT applications planned for the future.
Efficient Resource Utilization:
Avoids overloading either the edge or the cloud layer to an extent that the other is not relied on enough.Improves scalability and reliability of the solution by guaranteeing accurate usage of computational resources.
Adaptability:
With this, the hybrid system provides an improved execution of the process for both large and small tasks and different resource demand rates.
From above simulation results, it can be asserted that, the proposed hybrid edge-cloud architecture is efficient. The architecture also shows lower latency than edge-only and cloud-only platforms but does not overload the layers at the edges of the network. Thus, the results presented in this paper show that using the hybrid system is effective for real-time and IoT applications of large dimension.

Conclusion

This research work formulated and tested an edge-cloud integration framework with the aim of minimizing end-to-end delay and resource consumption for IoT use cases. The combination of the layered edge and cloud architecture effectively allowed the hybrid system to overcome the inherent weaknesses found in edge only and cloud only models. The simulations carried out on twenty different scenarios, using the tests bed of one hundred internet of Things devices in the hybrid architecture showed substantial increase in response time and use of resources which makes the proposed hybrid architecture a suitable model for current distributed cloud computing systems.
The latency analysis further showed that the proposed hybrid system obtained the least latency for all the different tasks sizes as it was able to manage the real time tasks on the edge and the batch tasks on the cloud. This strategic distribution of tasks helped to use cloud resources more efficiently, reducing data transmission time, while many complicated and acceptable for certain amount of latency tasks were performed on cloud resources. Hence, compared to the fully edge-only systems, which suffer from resource constraints leading to latency bottlenecks, the resulting system presented in the paper demonstrated reduced transmission delays, which are characteristic of fully cloud-based systems that negatively impact real-time applicability.
Another indication of the organizational efficiency was provided by the analysis of resources distribution when comparing the hybrid model with the types of workloads. The edge-only configuration was utilizing edge resources to the near-brink but saw cloud resources largely left idle. On the other hand, cloud-only configuration saturated cloud resources and left edge resources unutilized. On the other hand, compared to the basic system, the hybrid system coordinated in the manner of the dynamic distribution of the tasks between the two layers; the overall resource consumption was 70% at the edge layer and 75% at the cloud layer. This balance not only maintained the system scale, but also made the system run more stably and efficiently under various work loads.
Core attributes like real-time data acquisition, task scheduling, dynamic assignment of resources, and analytical tools were implemented correctly into a theoretical model by the proposed framework. Using AnyLogic for simulation, Siemens Tecnomatix and MATLAB presented insights into the effectiveness of the proposed architecture. It also confirmed the system's capability to adjust the execution to fluctuating

workloads, allocate resources effectively, and deliver responses promptly, which points towards the extensive use of the system in a plethora of IoT domains, including smart cities, smart industry, and others.
In total, this work can be seen as the systematic approach towards efficient and scalable implementation of the edge-cloud systems. It solves issues of high latency, over-provisioning of resources, and systems flexibility to allow for the development of the next-generation distributed computing solutions. Herein, this paper presents the hybrid edge-cloud architecture with a better latency and resource utilization compared to the decentralized and centralized IoT architecture and confirms this architecture as a viable solution for real-time IoT applications.
Some of the future works can be targeted to apply the hand machine learning-based scheduling algorithm to improve the utilisation of resources and the prediction of task delays. This would make distributed workloads at the edge and cloud layers even more flexibly and efficiently manageable in terms of complexity and dynamics.

References

1. Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, August). Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing (pp. 13-16).
2. Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4), 14-23.
3. Sarkar, S., & Misra, S. (2016). Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. Iet Networks, 5(2), 23-29.
4. Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. International journal of communication systems, 25(9), 1101.
5. Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18), 1587-1611.
6. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.
7. Barroso, L. A., & Clidaras, J. (2022). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Springer Nature.
8. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.
9. Flinn, J. (2022). Cyber foraging: Bridging mobile and cloud computing. Springer Nature.
10. Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1, 7-18.
11. Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. Computer networks, 57(10), 2266-2279.
12. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.
13. Wang, L., Von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., & Fu, C. (2010). Cloud computing: a perspective study. New generation computing, 28, 137-146.

14. Cardellini, V., Colajanni, M., & Yu, P. S. (2000, August). Geographic load balancing for scalable distributed web systems. In Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No. PR00728) (pp. 20-27). IEEE.
15. Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. International journal of communication systems, 25(9), 1101.
16. Stojmenovic, I., & Wen, S. (2014, September). The fog computing paradigm: Scenarios and security issues. In 2014 federated conference on computer science and information systems (pp. 1-8). IEEE.
17. Ma, H., Chan, K. W., & Liu, M. (2013). An intelligent control scheme to support voltage of smart power systems. IEEE transactions on industrial informatics, 9(3), 1405-1414.
18. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. IEEE communications surveys & tutorials, 16(1), 414-454.
19. Linthicum, D. S. (2009). Cloud computing and SOA convergence in your enterprise: a step-by-step guide. Pearson Education.
20. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile networks and applications, 19, 171-209.
21. Poulis, A., Panigyrakis, G., & Panos Panopoulos, A. (2013). Antecedents and consequents of brand managers' role. Marketing Intelligence & Planning, 31(6), 654-673.
22. Shilpa, Lalitha, Prakash, A., & Rao, S. (2009). BFHI in a tertiary care hospital: Does being Baby friendly affect lactation success?. The Indian Journal of Pediatrics, 76, 655-657.
23. Gopinath, S., Janga, K. C., Greenberg, S., & Sharma, S. K. (2013). Tolvaptan in the treatment of acute hyponatremia associated with acute kidney injury. Case reports in nephrology, 2013(1), 801575.
24. Swarnagowri, B. N., & Gopinath, S. (2013). Ambiguity in diagnosing esthesioneuroblastoma--a case report. Journal of Evolution of Medical and Dental Sciences, 2(43), 8251-8255.
25. Malhotra, I., Gopinath, S., Janga, K. C., Greenberg, S., Sharma, S. K., & Tarkovsky, R. (2014). Unpredictable nature of tolvaptan in treatment of hypervolemic hyponatremia: case review on role of vaptans. Case reports in endocrinology, 2014(1), 807054.
26. Swarnagowri, B. N., & Gopinath, S. (2013). Pelvic Actinomycosis Mimicking Malignancy: A Case Report. tuberculosis, 14, 15.
27. Shakibaie-M, B. (2008). Microscope-guided external sinus floor elevation (MGES)–a new minimally invasive surgical technique. IMPLANTOLOGIE, 16(1), 21-31.
28. Vozikis, A., Panagiotou, A., & Karakolias, S. (2021). A Tool for Litigation Risk Analysis for Medical Liability Cases. HAPSc Policy Briefs Series, 2(2), 268-277.
29. Shakibaie-M, B. (2010). Uses of the operating microscope in minimally invasive implantology. Quintessenz, 61(3), 293-308.
30. Strietzel, F. P., & Shakibaie, B. (1998). Der Einsatz der TefGen-FD-Membran zum Erhalt des Alveolarkamms nach Zahnextraktionen. Deutsche Zahnärztliche Zeitschrift, 53(12), 883-886.
31. Shakibaie-M, B. (2010). Uses of the operating microscope in minimally invasive implantology. Quintessenz, 61(3), 293-308.