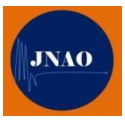
SK Khaja Mohinoddin, Sunil Gupta, J. Nonlinear Anal. Optim. Vol. 17(1) (2026),
Journal of Nonlinear Analysis and Optimization
Volume 17(1) (2026) ISSN : 1906-9685 (print) ISSN : 3057-0867 (online)
https://ph03.tci-thaijjo.org/index.php/jnao



J. Nonlinear Anal. Optim

Energy-Efficient Cloud-Edge Scheduling with QoS Preservation: Simulation and Testbed-Based Evaluation

Shaik Khaja Mohinoddin[1], Sunil Gupta[2]

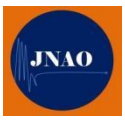
[1] Research Scholar, Department of Computer Science Engineering, Sikkim Alpine University, Sikkim, India, 737126

[2] Professor, Department of Computer Science Engineering, Sikkim Alpine University, Sikkim, India, 737126

Abstract

Energy-efficient and QoS-preserving scheduling is a major requirement in cloud–edge computing environments, where heterogeneous resources, dynamic workloads, and distributed data sources create complex performance–energy trade-offs. Although existing orchestration platforms provide automated deployment, scheduling, scaling, and workload management, many of them remain compute-centric and reactive. They commonly prioritize CPU availability, memory capacity, and load balancing while providing limited consideration for energy consumption, network latency, data movement cost, and service-level agreement satisfaction. As a result, conventional scheduling strategies may lead to excessive energy usage, higher latency, poor resource utilization, and increased SLA violations in cloud–edge infrastructures. This paper presents an experimental evaluation of an energy-efficient cloud–edge scheduling approach with QoS preservation. The proposed scheduling model supports intelligent task placement, workload scheduling, auto-scaling, VM/container migration, and workload consolidation across heterogeneous cloud, fog, and edge resources. The model integrates compute suitability, network condition, data locality, workload prediction, and Energy–QoS optimization to generate adaptive orchestration decisions. Unlike baseline schedulers that optimize isolated objectives, the proposed approach jointly minimizes energy consumption, latency, makespan, data transfer cost, and QoS violation while improving resource utilization and throughput. The performance of the proposed scheduling model is evaluated using multiple metrics, including total energy consumption, energy saving, average latency, makespan, SLA satisfaction rate, SLA violation rate, resource utilization, scheduling overhead, throughput, and data transfer cost reduction. The proposed model is compared with three related scheduling approaches: default Kubernetes scheduling, energy-aware scheduling, and ML-based resource scheduling. Experimental results show that the proposed approach achieves 32.15% energy saving, reduces average latency to 41.60 ms, improves SLA satisfaction to 98.10%, and increases resource utilization to 90.30%. Ablation analysis further confirms that task preprocessing, context modeling, workload prediction, multi-criteria decision-making, Energy–QoS optimization, and feedback learning each contribute progressively to the final improvement. Statistical significance testing validates that the observed gains are consistent and meaningful across repeated experimental runs.

Keywords: Cloud–edge scheduling; energy efficiency; Energy–QoS optimization; QoS preservation; resource orchestration; SLA satisfaction; task placement; workload consolidation; workload prediction; sustainable cloud computing.



1. Introduction

The increasing adoption of cloud–edge computing has changed the way computational workloads are deployed, scheduled, and executed across distributed infrastructures. Traditional cloud computing offers large-scale processing, storage, and virtualization capabilities; however, many modern applications such as IoT monitoring, autonomous systems, healthcare analytics, video surveillance, industrial automation, and smart-city services require faster response time and localized processing. Edge computing addresses this requirement by placing computation closer to data sources, while the cloud continues to provide high-capacity processing for computation-intensive workloads. As a result, modern distributed systems are no longer limited to centralized data centers but operate across a cloud–edge continuum, where cloud servers, fog nodes, edge gateways, and end devices must be coordinated together.

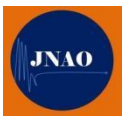
In such an environment, task scheduling becomes a critical challenge because the scheduler must decide where each workload should execute while considering computation demand, network delay, data movement, node capacity, energy consumption, and service-level constraints. Edge computing surveys have shown that resource scheduling generally involves computation offloading, resource allocation, and resource provisioning, where each decision affects latency, energy consumption, and service quality [1]. Recent studies further highlight that scheduling across the computing continuum must consider placement, energy efficiency, heterogeneous infrastructure, and distributed application requirements [2]. Therefore, cloud–edge scheduling cannot be treated as a simple node-selection problem; instead, it must be addressed as a multi-objective decision-making problem.

Energy efficiency is one of the most important concerns in cloud–edge scheduling. Cloud data centers consume substantial energy due to continuous workload execution, cooling, and infrastructure operation. At the same time, edge devices often operate under constrained power conditions. Therefore, an inefficient scheduling decision can increase both operational cost and environmental impact. Recent studies on energy-aware edge computing indicate that energy optimization must be addressed at different levels, including device-level execution, task offloading, middleware, application placement, and system-level resource management [3]. However, energy minimization alone is not sufficient because aggressive energy-saving strategies may degrade QoS, increase response time, and cause deadline violations.

Quality of Service preservation is equally important in cloud–edge environments. Many cloud–edge applications are latency-sensitive and require strict response-time guarantees. QoS-aware orchestration studies have emphasized that applications in the cloud–edge continuum must be dynamically deployed and adapted according to changing service requirements [4]. Serverless and function-based cloud–edge systems further demonstrate that heterogeneous functions and service classes require QoS-aware offloading policies to maintain service utility under resource and budget constraints [5]. Hence, an effective scheduler must balance energy efficiency with QoS preservation rather than optimizing either objective independently.

Several recent studies have proposed energy-efficient and QoS-aware scheduling methods for cloud, edge, and hybrid environments. Deep reinforcement learning-based task scheduling has been used to improve decision-making in cloud computing [6]. Other approaches use multi-criteria decision-making for energy-efficient cooperative resource allocation in IoT edge environments [7]. Energy-efficient resource allocation and management strategies have also been proposed for improving resource usage and reducing power consumption in fog-enabled systems [8]. These works show that intelligent and optimization-based scheduling can improve resource management; however, many of them still focus on a limited set of objectives or specific environments.

Another important limitation in existing scheduling approaches is the incomplete integration of network and data movement costs. In cloud–edge systems, data is often generated at edge devices,



but computation may be executed at fog nodes or cloud data centers. If the scheduler selects a distant compute node only because it has high processing capacity, the system may suffer from increased latency, bandwidth consumption, and energy cost due to data transfer. Recent works on integrated computing, caching, and communication in the cloud-to-edge continuum emphasize the need to jointly consider computation, communication, and data-related decisions [9].

Adaptive scheduling is also necessary because cloud–edge workloads are highly dynamic. Workload arrival rates, user mobility, network congestion, node availability, and energy status can change continuously. Static scheduling policies and rule-based heuristics may perform well under fixed conditions but fail when the system state changes. Recent research on intelligent job scheduling and real-time resource optimization for edge-cloud continuum systems demonstrates the importance of adaptive decision-making in next-generation networks [10,11]. Autonomous energy-aware scheduling mechanisms for serverless edge environments also show that dynamic scheduler selection can improve energy-aware function execution [12,13].

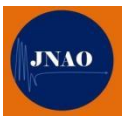
Despite this progress, several research limitations remain. First, many existing schedulers remain compute-centric and consider CPU, memory, or task load as the primary decision factors. Second, some energy-aware methods improve energy saving but do not sufficiently preserve QoS under latency-sensitive workloads. Third, many QoS-aware methods reduce response time but do not explicitly minimize energy consumption and data transfer cost. Fourth, several ML-based methods focus on prediction or offloading but do not provide a complete scheduling pipeline that includes placement, migration, scaling, workload consolidation, and feedback-based policy adaptation.

Motivated by these limitations, this paper presents an Energy-Efficient Cloud–Edge Scheduling Model with QoS Preservation. The proposed model supports task placement, workload scheduling, auto-scaling, VM/container migration, and workload consolidation across heterogeneous cloud, fog, and edge resources. Unlike conventional schedulers, the proposed approach jointly considers compute suitability, network condition, data locality, workload prediction, energy consumption, and QoS constraints. The scheduling problem is formulated as a multi-objective optimization process that minimizes energy consumption, latency, makespan, data transfer cost, and QoS violation while improving SLA satisfaction, throughput, and resource utilization.

The major contributions of this paper are summarized as follows:

1. An energy-efficient and QoS-preserving cloud–edge scheduling model is presented for heterogeneous distributed infrastructures.
2. The proposed model integrates compute suitability, network awareness, data locality, workload prediction, and Energy–QoS optimization into a unified scheduling process.
3. A multi-dimensional evaluation framework is used to assess total energy consumption, energy saving, average latency, makespan, SLA satisfaction, SLA violation rate, resource utilization, scheduling overhead, throughput, and data transfer cost reduction.
4. The proposed model is compared with default Kubernetes scheduling, energy-aware scheduling, ML-based scheduling, and selected existing methods.
1. Ablation analysis and statistical significance testing are included to validate the contribution of each component and confirm the reliability of performance improvements

The rest of the paper is organized as follows. Section 2 reviews existing studies on energy-aware scheduling, DL and RL-based scheduling, serverless cloud edge scheduling and multi-objective optimization. Section 3 identifies the key research gaps and explains how the proposed framework addresses them. Section 4 presents the proposed methodology. Section 5 discusses the experimental



results, baseline comparisons, ablation study, and statistical significance analysis. Finally, Section 6 concludes the paper and outlines future research directions.

2. Literature Survey

Energy-efficient scheduling in cloud–edge environments has become an important research direction because modern applications increasingly require low latency, high QoS, reduced energy consumption, and efficient resource utilization. Existing studies have explored energy-aware scheduling, QoS-aware task allocation, serverless offloading, reinforcement learning-based scheduling, and sustainable edge-cloud workload distribution. These works demonstrate that energy, latency, resource availability, workload priority, and service requirements must be considered during task placement and resource allocation. However, most existing methods focus on specific environments such as vehicular edge computing, IoT-edge systems, mobile edge computing, or serverless workflows. As a result, they often do not provide a unified scheduling and orchestration framework that can jointly handle compute suitability, network condition, data locality, workload prediction, Energy–QoS optimization, task placement, scaling, migration, consolidation, and feedback-based adaptation across heterogeneous cloud–fog–edge infrastructures.

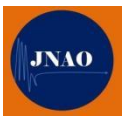
2.1 Energy-Efficient Scheduling in Cloud-Edge Environments

support computation-intensive workloads while reducing power consumption and operational cost. In [14], cloud–edge data center optimization was studied by jointly considering security, QoS, and energy efficiency. The study highlights the importance of balancing energy consumption with service requirements; however, it focuses mainly on selected optimization dimensions and does not fully integrate placement, scaling, migration, workload consolidation, and feedback-based scheduling into one unified framework.

QoS-aware task scheduling has also been explored in vehicular edge computing. In [15], joint task scheduling and resource allocation were formulated to improve QoS under vehicular edge conditions. The method demonstrates the value of considering both computation and communication resources during scheduling. However, such domain-specific models are often designed for vehicular scenarios and may not generalize directly to heterogeneous cloud–fog–edge infrastructures.

In [16], energy-efficient cooperative resource allocation and task scheduling were investigated for IoT edge environments using a multi-criteria decision-making strategy. The work shows that ranking-based resource selection can improve energy efficiency and task allocation. However, predefined criteria and fixed weights may reduce adaptability under dynamic workload, network, and energy conditions.

Similarly, [17] proposed an efficient resource allocation and management strategy with energy efficiency for fog-enabled systems. Although it improves energy-aware resource assignment, it mainly focuses on allocation efficiency and does not provide a complete cloud–edge scheduling pipeline involving network awareness, data locality, workload prediction, Energy–QoS optimization, and adaptive decision enforcement.



2.2 Deep Learning and Reinforcement Learning-Based Scheduling

Deep reinforcement learning-based scheduling has been examined in [18], where an energy-efficient task scheduler was designed for cloud computing. The approach learns scheduling decisions through interaction with the environment and improves scheduling performance compared with conventional reinforcement learning methods. However, such models often require careful reward design and may not explicitly include data locality, communication cost, and SLA violation penalties in the scheduling objective.

Task offloading and resource allocation in edge computing networks were studied in [19], where energy consumption, latency, task heterogeneity, and resource constraints were considered. This confirms that scheduling in edge systems requires joint optimization of multiple conflicting objectives. However, many such approaches focus mainly on offloading rather than a complete scheduling workflow covering placement, scaling, migration, consolidation, and feedback-driven adaptation.

In [20], an energy-efficient and priority-aware task scheduling model was proposed for dynamic vehicular edge computing. The model highlights the role of task priority in scheduling decisions. However, priority-aware scheduling must be further integrated with energy–QoS balancing and data-locality awareness to support general cloud–edge environments.

In [21], task offloading optimization was performed using density clustering, ensemble learning, and deep reinforcement learning. The method improves offloading decision-making in mobile edge computing. However, ensemble-based scheduling can increase model complexity and decision overhead, which may affect real-time orchestration in large-scale cloud–edge systems.

2.3 Serverless and Function-Level Cloud-Edge Scheduling

Serverless cloud–edge scheduling has also received attention. In [22], QoS-aware offloading policies were developed for serverless functions in the cloud-to-edge continuum. The study shows that function heterogeneity and service classes must be considered during scheduling. However, serverless-specific methods may not directly address broader scheduling scenarios involving containers, virtual machines, microservices, and data-intensive workloads.

Energy-aware serverless scheduling was addressed in [23], where scheduler selection was adapted according to energy conditions at edge nodes. This demonstrates the importance of energy-awareness in edge execution. However, scheduler selection alone may not fully optimize network latency, data locality, SLA deadlines, and workload consolidation.

In [24], an autonomous energy-aware scheduling mechanism was proposed for serverless IoT edge environments. The work emphasizes residual energy-aware resource allocation. However, residual-energy-based scheduling requires extension with compute suitability, network status, and QoS-aware optimization to support heterogeneous cloud–edge infrastructures.

In [25], energy-aware serverless workflow orchestration was studied for the computing continuum. The work considers energy consumption and sustainability in workflow orchestration. However,

serverless workflow optimization differs from general-purpose cloud–edge scheduling, where workloads may include containerized services, VM-based tasks, and mixed application types.

2.4 Task Offloading, Resource Allocation, and Data Movement

Task offloading and resource allocation were further studied in [26], where latency and energy consumption were optimized in mobile edge computing. This confirms that offloading must balance execution delay and power usage. However, many offloading approaches still focus mainly on device-to-edge allocation and do not fully represent hierarchical cloud–fog–edge execution.

In [27], cloud–edge–end cooperative task offloading and resource allocation were formulated using a time-slotted scheduling model. The study supports cooperation across different infrastructure layers. However, time-slotted formulations may be less flexible under unpredictable workload arrivals, changing link conditions, and real-time SLA constraints.

Distributed task offloading and resource allocation were investigated in [28], where latency and energy trade-offs were considered. Distributed scheduling can improve scalability and reduce centralized control dependency. However, it may also introduce coordination overhead and may not always capture global Energy–QoS optimization across the entire cloud–edge continuum.

In [29], task offloading for energy-harvesting IoT devices was studied by balancing execution utility and cost. The work is useful for energy-constrained IoT scenarios. However, a broader evaluation is required for general cloud–edge scheduling, including SLA satisfaction, resource utilization, makespan, throughput, scheduling overhead, and statistical significance.

2.5 Multi-Objective Optimization and Sustainable Scheduling

Multi-objective scheduling has been addressed in [30], where task offloading and resource allocation were jointly optimized in MEC-enabled medical vehicular networks. The study confirms the importance of optimizing completion time and energy consumption together. However, application-specific models require generalization before being applied to heterogeneous cloud–edge infrastructures.

Sustainable edge-cloud data center management was discussed in [31], where energy efficiency and sustainability were emphasized for next-generation edge-cloud data centers. This supports the need for scheduling models that evaluate both performance and sustainability. However, sustainability-oriented scheduling must still be integrated with practical orchestration actions such as task placement, migration, scaling, and consolidation.

Low-latency and low-carbon workload distribution was studied in [32], where edge-cloud collaboration was used to support mixed workloads. The work highlights the value of balancing latency and sustainability. However, low-carbon collaboration models still require stronger integration with QoS preservation, data movement reduction, and adaptive scheduling policies.

In [33], QoS-aware placement of interdependent services was investigated in energy-harvesting edge computing environments. The work considers service dependency and energy uncertainty. However, the present paper extends this direction by integrating compute suitability, network condition, data



locality, workload prediction, and Energy–QoS optimization into a unified cloud–edge scheduling model.

3. Research Gaps and Gap-Addressing Strategy

From the reviewed literature, it is observed that energy-aware scheduling, QoS-driven task allocation, edge computing, serverless orchestration, machine learning-based resource management, and multi-objective optimization have individually contributed useful solutions for cloud–edge environments. However, these approaches remain partially fragmented because they often address only selected aspects such as energy reduction, latency minimization, task offloading, or resource allocation. They rarely provide a unified orchestration mechanism that jointly considers compute suitability, network condition, data locality, workload prediction, Energy–QoS trade-off, SLA preservation, and adaptive feedback-based decision-making. Therefore, the proposed study identifies these limitations and addresses them through a unified multi-perspective cloud–edge orchestration framework designed to support sustainable, adaptive, and QoS-aware resource scheduling.

1. ***Fragmented objective optimization:*** *Many existing scheduling approaches optimize only selected objectives such as energy consumption, latency, or task offloading. However, limited studies integrate energy, QoS, SLA violation, resource utilization, communication cost, data locality, and scheduling overhead into a single unified decision model.*

2. ***Limited generalization across cloud-edge infrastructures:*** *Several existing methods are designed for specific domains such as vehicular edge computing, IoT systems, mobile edge computing, or serverless platforms. This limits their applicability to heterogeneous cloud-fog-edge infrastructures.*

3. ***Incomplete orchestration coverage:*** *Most approaches focus mainly on task offloading or resource allocation. They do not fully address practical orchestration actions such as task placement, scaling, migration, workload consolidation, and adaptive decision enforcement.*

4. ***Weak integration of data locality and network awareness:*** *Data locality and network conditions are not sufficiently considered in many scheduling models. This is a major limitation because data movement and communication latency directly affect energy consumption and QoS in cloud–edge environments.*

5. ***Static decision policies:*** *Many methods depend on fixed criteria, predefined weights, or static decision rules. Such approaches reduce adaptability under dynamic workloads, changing network conditions, and varying energy availability.*

6. ***Limited feedback-driven adaptation:*** *Feedback-based scheduling and continuous adaptation remain underexplored. Many existing models do not update future scheduling decisions using observed execution outcomes, SLA deviations, energy behavior, and workload variations.*

The proposed framework addresses these gaps by introducing a unified Energy–QoS-aware scheduling and resource orchestration strategy for heterogeneous cloud–edge environments. To overcome fragmented objective modeling, the framework jointly considers energy consumption, latency, SLA violation, resource utilization, communication cost, data locality, and workload demand during scheduling decisions. To improve generalization beyond domain-specific systems, the framework is designed for cloud–fog–edge infrastructures involving diverse execution nodes, workload types, and

resource capacities. To move beyond simple task offloading, the proposed model incorporates multiple orchestration actions, including task placement, scaling, migration, workload consolidation, and adaptive decision enforcement. Network condition and data locality are integrated into the decision process so that computation can be placed closer to relevant data sources when beneficial, thereby reducing communication cost and improving QoS. The framework also includes workload prediction and context analysis to support proactive scheduling instead of purely reactive allocation. Furthermore, a feedback loop continuously monitors execution outcomes, energy usage, latency, SLA status, and resource utilization, allowing future decisions to be refined dynamically. In this way, the proposed framework provides a more adaptive, practical, and comprehensive solution for sustainable cloud–edge scheduling.

4. Proposed Methodology

The proposed methodology presents an Energy–QoS Aware Cloud–Edge Scheduling and Resource Orchestration Model for improving energy efficiency, latency performance, SLA satisfaction, and resource utilization across heterogeneous cloud, fog, and edge infrastructures. The model evaluates each task using both workload-specific and infrastructure-specific conditions before making a scheduling decision. The architecture consists of two major proposed components: Component 1: Intelligent Task Analysis and Context Modeling and Component 2: Energy–QoS Aware Scheduling and Resource Orchestration. Component 1 transforms incoming tasks and system telemetry into structured scheduling context, while Component 2 uses this context to make optimized scheduling decisions and enforce orchestration actions.

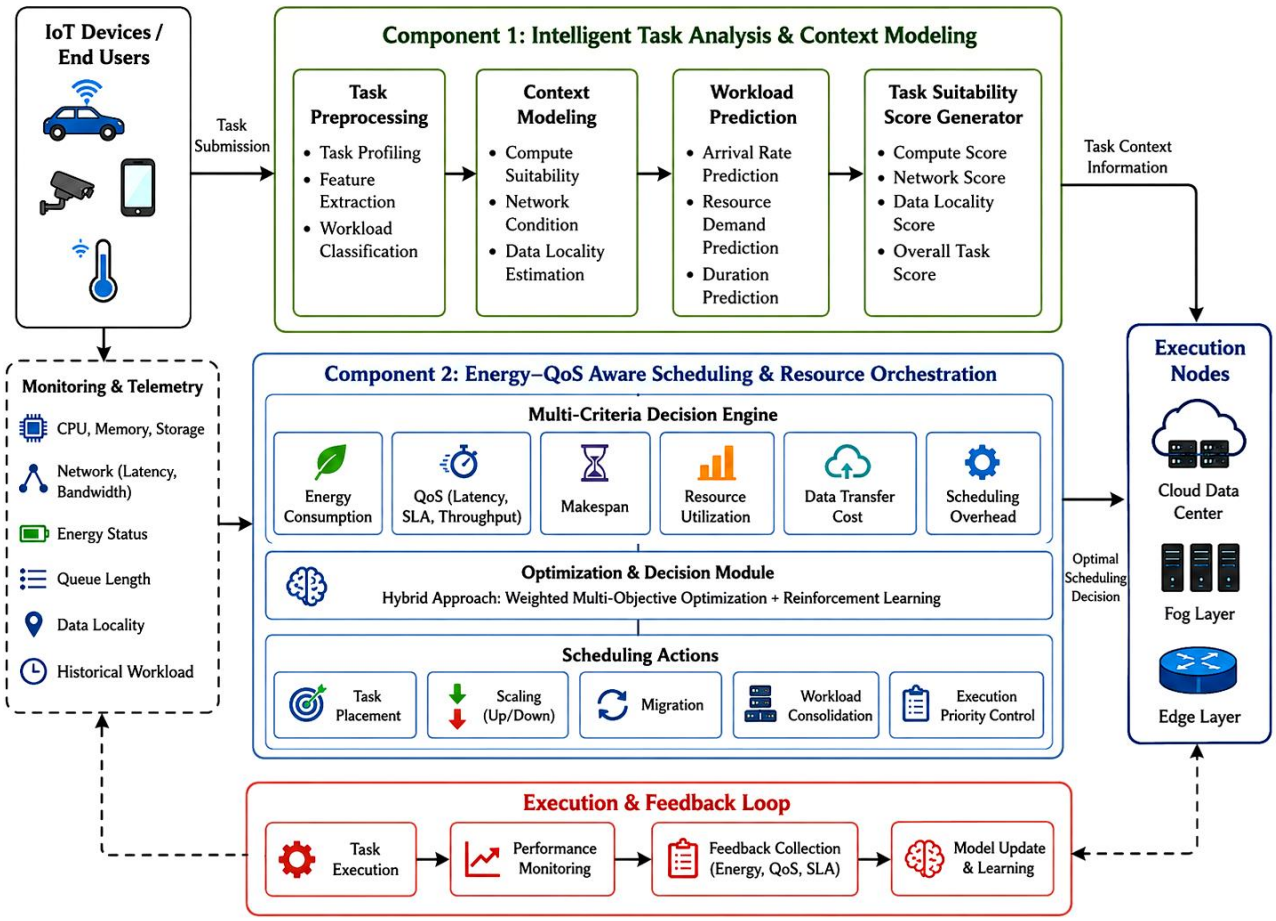
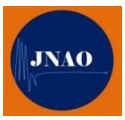


Figure 1. Architectural diagram of the proposed Energy-QoS aware cloud-edge scheduling model.

Figure 1 presents the overall architecture of the proposed energy-QoS-aware cloud-edge scheduling and resource orchestration framework. The framework begins with task submissions from IoT devices and end users, while monitoring and telemetry data are continuously collected from the infrastructure, including CPU, memory, storage, network condition, energy status, queue length, data locality, and historical workload. The first component, Intelligent Task Analysis and Context Modeling, processes the submitted tasks by performing task profiling, feature extraction, workload classification, context modeling, workload prediction, and task suitability scoring. This generates task-context information based on compute suitability, network condition, data locality, and predicted workload demand. The second component, Energy-QoS Aware Scheduling and Resource Orchestration, uses this task-context information to make optimal scheduling decisions. It evaluates multiple decision criteria, including energy consumption, QoS parameters, makespan, resource utilization, data transfer cost, and scheduling overhead. These criteria are then processed through an optimization and decision module that combines weighted multi-objective optimization with reinforcement learning. Based on the decision outcome, the framework performs scheduling actions such as task placement, scaling, migration, workload consolidation, and execution priority control



across cloud data centers, fog layers, and edge layers. The execution and feedback loop continuously monitors the performance of scheduled tasks after deployment. It collects feedback related to energy usage, QoS performance, and SLA status, and uses this information to update the decision model. This feedback-driven mechanism enables the framework to adapt to changing workload, network, and energy conditions. Therefore, Figure 1 highlights how the proposed framework integrates task analysis, context modeling, multi-criteria optimization, scheduling enforcement, and continuous learning into a unified cloud–edge orchestration architecture.

4.1 System Model

Let the cloud–edge infrastructure be represented as a set of heterogeneous execution nodes: $N = \{n_1, n_2, n_3, \ldots, n_m\}$, where each node may belong to the cloud data center, fog layer, or edge layer. Each node is characterized by processing capacity, memory capacity, bandwidth availability, energy profile, and current utilization status: $n_j = \{C_j, M_j, B_j, E_j, U_j\}$. The workload submitted to the system is represented as:$T = \{t_1, t_2, t_3, \ldots, t_k\}$. Each task is described by computation demand, memory requirement, input data size, deadline, priority, and QoS sensitivity: $t_i = \{c_i, m_i, d_i, \delta_i, p_i, q_i\}$

The scheduling decision is represented using a binary assignment variable as in Eqn. 1:

$$x_{ij} = 1 \text{ if task ti is assigned to node } n_j \text{; otherwise } x_{ij} = 0 \quad , \tag{1}$$

where, $\Sigma_j x_{ij} = 1, \quad \forall t_i \in T$

4.2 Component 1: Intelligent Task Analysis and Context Modeling

Figure 2 illustrates the first major component of the proposed framework, namely Intelligent Task Analysis and Context Modeling. This component converts raw workload requests and infrastructure-level monitoring information into a structured task-context output that can be directly used by the scheduler. Incoming workloads may originate from IoT devices, mobile users, surveillance systems, healthcare sensors, or enterprise applications. Each workload is associated with different characteristics such as computation demand, memory requirement, input data size, deadline, priority, and workload category. Therefore, before scheduling decisions are made, the system first analyzes both task-level requirements and infrastructure-level conditions.

The first stage is task preprocessing, where each incoming task is profiled, features are extracted, and the workload is classified according to its execution characteristics. Let a task be represented as: $T_i = C_i, M_i, D_i, L_i, P_i, W_i$, where C_i denotes computation demand, M_i represents memory requirement, D_i indicates input data size, L_i denotes deadline or latency requirement, P_i represents task priority, and $W_i$denotes workload category. This representation allows the framework to understand the basic requirements of each task before it is forwarded to the scheduling layer.

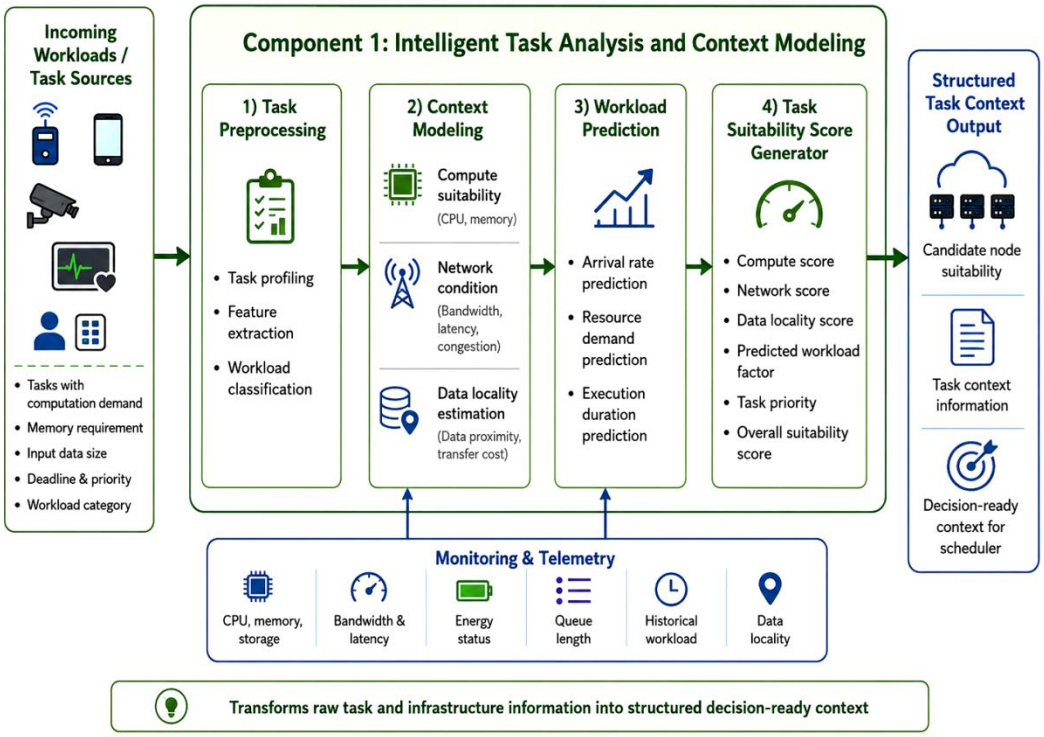


Figure 2.. Intelligent Task Analysis and Context Modeling Component of the Proposed Framework

The second stage is context modeling, where infrastructure conditions are analyzed using monitoring and telemetry data. The framework considers compute availability, network condition, and data locality. Compute suitability is estimated using the available CPU and memory resources of a candidate node. It can be expressed as in Eqn. 2:

$$CS_{ij} = \alpha_1 CPU_j + \alpha_2 MEM_j, \tag{2}$$

where CS_{ij} is the compute suitability score of task T_i on node N_j, CPU_j and MEM_j represent available CPU and memory capacity of node N_j, and α_1 and α_2 are weighting factors. A higher compute suitability score indicates that the candidate node is more capable of executing the task efficiently.

The third stage performs workload prediction using historical and real-time telemetry information. This stage estimates task arrival rate, future resource demand, and expected execution duration. The predicted workload demand can be represented as mentioned in Eqn. 3:

$$\widehat{W}_{t+1} = f(W_t, W_{t-1}, Q_t, R_t) \tag{3}$$

where $\widehat{W}_{t+1}$ is the predicted workload for the next time interval, W_t and W_{t-1} are current and previous workload states, Q_t is the current queue length, and R_t represents current resource utilization. This prediction helps the scheduler make proactive decisions instead of responding only after resource overload or SLA degradation occurs.

The fourth stage is the task suitability score generator, which combines compute score, network score, data locality score, predicted workload factor, and task priority into a single overall suitability value. The overall suitability score can be expressed as in Eqn. 4:

$$OSS_{ij} = \beta_1 CS_{ij} + \beta_2 NS_{ij} + \beta_3 DLS_{ij} + \beta_4 PWF_i + \beta_5 P_i \quad (4)$$

where OSS_{ij} denotes the overall suitability score of task T_i for node N_j, NS_{ij} is the network score, DLS_{ij} is the data locality score, PWF_i is the predicted workload factor, and P_i is the task priority. The coefficients β_1 to β_5 define the relative importance of each factor.

The final output of this component is a structured task-context representation consisting of candidate node suitability, task-context information, and decision-ready scheduling input. Thus, Figure 2 shows how the proposed framework transforms raw task and infrastructure data into meaningful scheduling intelligence. This enables the next component of the framework to perform more accurate Energy-QoS-aware scheduling and resource orchestration across cloud, fog, and edge layers.

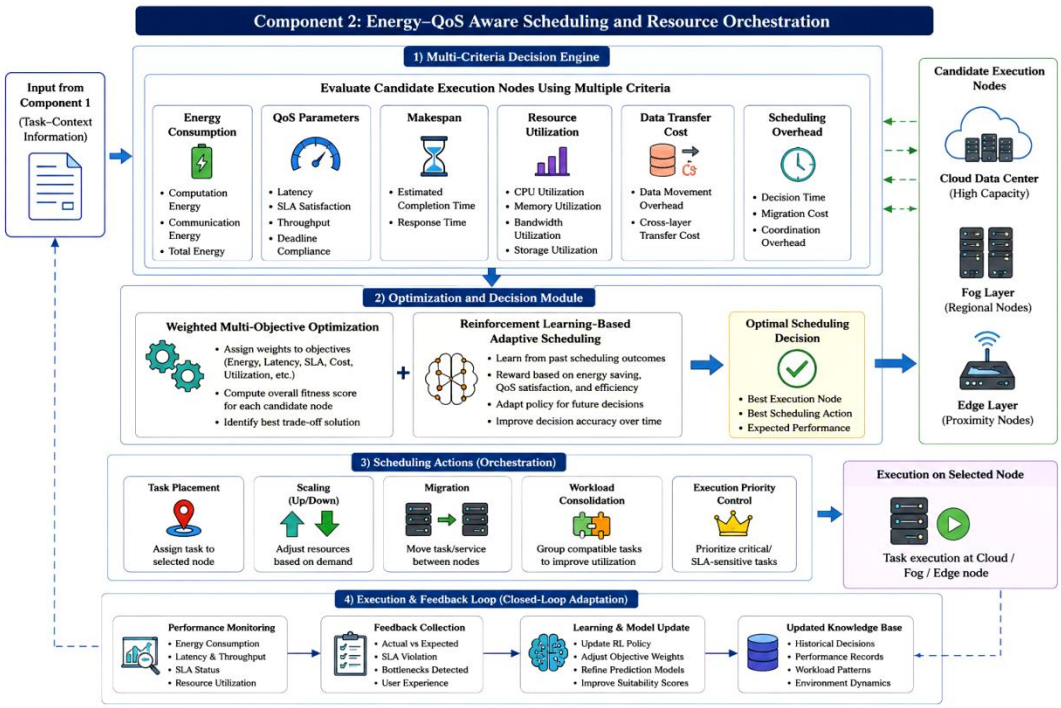


Figure 3. Component 2: Energy–QoS Aware Scheduling and Resource Orchestration.

4.3 Component 2: Energy-QoS Aware Scheduling and Resource Orchestration

Figure 3 presents the second major component of the proposed framework, namely Energy–QoS-Aware Scheduling and Resource Orchestration. This component receives structured task-context information from Component 1 and uses it to select the most suitable execution node across cloud data centers, fog nodes, and edge devices. The purpose of this component is to make scheduling decisions that



balance energy efficiency, QoS preservation, resource utilization, data transfer cost, and scheduling overhead.

The first stage is the multi-criteria decision engine, where candidate execution nodes are evaluated using multiple scheduling criteria. These include energy consumption, QoS parameters, makespan, resource utilization, data transfer cost, and scheduling overhead. Energy consumption is estimated by considering computation energy and communication energy as in Eqn. 5:

$$E_{total} = E_{comp} + E_{comm}, \tag{5}$$

where E_{total} represents the total energy required for task execution, E_{comp} is the energy consumed during computation, and E_{comm} is the energy consumed during data transmission. This helps the framework avoid scheduling decisions that reduce latency but increase unnecessary energy cost.

The second stage is the optimization and decision module, which combines weighted multi-objective optimization with reinforcement learning-based adaptive scheduling. The weighted optimization component assigns importance to different objectives such as energy, latency, SLA cost, utilization, and transfer cost. The overall scheduling objective can be expressed as in Eqn. 6:

$$F_{ij} = w_1 E_{ij} + w_2 L_{ij} + w_3 SLA_{ij} + w_4 C_{ij} + w_5 O_{ij}, \tag{6}$$

where F_{ij} is the overall fitness cost of assigning task T_i to node N_j, E_{ij} is energy consumption, L_{ij} is latency, SLA_{ij} is SLA violation cost, C_{ij} is data transfer cost, O_{ij} is scheduling overhead, and $w_1$to w_5 are objective weights. The node with the lowest fitness cost is considered more suitable for task execution.

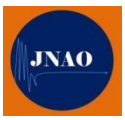
Reinforcement learning is used to improve scheduling decisions over time by learning from previous execution outcomes. After each scheduling action, the system observes performance feedback such as latency, throughput, SLA satisfaction, energy usage, and resource utilization. The reward function may be expressed as in Eqn. 7:

$$R = \lambda_1 QoS - \lambda_2 E_{total} - \lambda_3 SLA_{viol}\ , \tag{7}$$

where R is the reward value, (QoS) represents the achieved quality of service, E_{total} is the total energy onsumption, SLA_{viol} denotes SLA violation penalty, and λ_1, λ_2, and λ_3 control the importance of each term. A higher reward indicates that the scheduling action achieved better QoS with lower energy and fewer SLA violations.

Based on the optimized scheduling decision, the framework performs different orchestration actions. These include task placement, scaling resources up or down, migrating workloads between nodes, consolidating compatible workloads to improve utilization, and prioritizing critical or SLA-sensitive tasks. Once the best execution node is selected, the task is deployed at the cloud, fog, or edge layer depending on its context and system conditions.

The final stage is the execution and feedback loop, which enables closed-loop adaptation. During execution, the system monitors energy consumption, latency, throughput, SLA status, and resource utilization. The collected feedback is used to update decision weights, refine prediction models, improve suitability scores, and maintain a knowledge base of historical decisions and performance records. Thus, Figure 3 shows how the proposed framework converts task-context information into optimized, adaptive, and energy-QoS-aware scheduling decisions for heterogeneous cloud–edge environments.



4.4 Execution and Feedback Loop

After the task is assigned to the selected execution node, the proposed framework activates the execution and feedback loop to monitor the actual runtime behavior of the scheduled task. This loop observes key performance indicators such as actual energy consumption, latency, SLA status, throughput, resource utilization, and completion time. These monitored values are used to evaluate whether the scheduling decision has achieved the expected Energy–QoS performance.

The feedback information at time (t) can be represented as in Eqn. 8:

$$FB(t) = EC_{actual}, LAT_{actual}, SLA_{status}, RU_{actual}, TP_{actual} \quad , \tag{8}$$

where EC_{actual} denotes the actual energy consumption, LAT_{actual} represents the observed latency, SLA_{status} indicates whether the SLA requirement is satisfied or violated, RU_{actual} denotes actual resource utilization, and TP_{actual} represents achieved throughput.

The system then compares the actual performance with the expected performance estimated during scheduling. The deviation between actual and expected performance is calculated using Eqn. 9:

$$\Delta(t) = F_{actual}(t) - F_{expected}(t), \tag{9}$$

where $F_{actual}(t)$ represents the observed execution performance and $F_{expected}(t)$ denotes the predicted performance before task execution.

If the absolute deviation exceeds a predefined threshold τ, the framework triggers a model update. This update may adjust scheduling weights, refine suitability scores, update workload prediction parameters, or improve reinforcement learning decisions. The update condition is expressed as shown in Eqn. 10:

$$Update = \begin{cases} 1, & \text{if} |\Delta(t)| > \tau \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Here, (Update = 1) indicates that the model parameters should be revised, while (Update = 0) indicates that the current scheduling behavior is acceptable. Through this feedback-driven mechanism, the proposed framework continuously adapts to variations in workload demand, network condition, resource availability, and energy behavior. As a result, the system becomes more stable, adaptive, and capable of maintaining Energy–QoS balance in dynamic cloud–edge environments.

4.5 Evaluation Metrics

The performance of the proposed Energy-QoS-aware scheduling framework is evaluated using multiple metrics related to energy efficiency, latency, QoS preservation, resource usage, and throughput. These metrics help measure how effectively the proposed scheduler reduces energy consumption while maintaining service quality and improving resource utilization in a heterogeneous cloud-fog-edge environment.

The total energy consumption is computed by summing the energy consumed by all assigned tasks across all execution nodes using Eqn. 11:

$$EC_{total} = \sum_i \sum_j x_{ij} \, EC_{ij}, \tag{11}$$

where EC_{total} represents the total energy consumption, x_{ij} is a binary decision variable indicating whether task i is assigned to node j, and EC_{ij} denotes the energy consumed when task i is executed on node j.



The average latency is calculated as the mean completion or response time of all executed tasks using Eqn. 12:

$$LAT_{avg} = \frac{1}{k}\sum_i T_i \ , \tag{12}$$

where LAT_{avg} denotes the average latency, k is the total number of completed tasks, and T_i represents the execution or response time of task i.

The SLA satisfaction rate measures the percentage of tasks completed within their defined deadline or service constraint using Eqn. 13:

$$SLA_{rate} = \left(\frac{N_{satisfied}}{N_{total}}\right) \times 100, \tag{13}$$

where $N_{satisfied}$ is the number of tasks that satisfy SLA requirements, and N_{total} is the total number of submitted tasks.

The resource utilization is used to evaluate how efficiently available infrastructure resources are used during task execution using Eqn. 14:

$$RU = \left(\frac{\sum UsedResources}{\sum TotalResources}\right) \times 100 \tag{14}$$

where RU represents the resource utilization percentage, $\sum UsedResources$ denotes the total amount of resources consumed, and $\sum TotalResources$ denotes the total available resources.

The energy saving percentage compares the energy consumption of the proposed scheduler with the baseline scheduler: ES, where $EC_{baseline}$ is the energy consumption of the baseline method and $EC_{proposed}$ is the energy consumption of the proposed scheduler. A higher value of ES indicates better energy-saving capability of the proposed framework.

5. Results and Analysis

This section presents the experimental design, simulation configuration, workload characteristics, scheduling parameters, and model hyperparameters used to evaluate the proposed Energy-QoS-aware scheduling and resource orchestration framework. The objective of the evaluation is to examine how effectively the proposed scheduler balances energy consumption, latency, SLA satisfaction, resource utilization, makespan, throughput, data transfer cost, and scheduling overhead in a heterogeneous cloud-fog-edge environment. The experimental setup is designed to support repeatable evaluation by combining simulation-based workload modeling with testbed-style infrastructure parameterization. Each scheduling method is executed over ten independent runs, and the mean performance values are reported for comparison. The repeated-run results are further used for statistical significance testing to verify whether the improvements achieved by the proposed scheduler are consistent and meaningful.

5.1. Experimental Setup

Table 1 presents the overall experimental setup used for evaluating the proposed scheduling framework. The execution environment is configured as a Python-based scheduling simulator with CloudSim/EdgeCloudSim-style workload and network modeling. This configuration enables controlled experimentation over a heterogeneous cloud–fog–edge infrastructure consisting of cloud data centers, fog nodes, and edge devices. The cloud layer includes high-capacity servers with 32–64 vCPUs and 64–128 GB RAM, while the fog layer consists of regional nodes with medium-level computational

capacity. The edge layer includes proximity nodes with lower resource capacity but reduced communication delay.

Table 1. Experimental Setup and Simulation/Testbed Configuration

Parameter	Configuration / Value
Execution environment	Python-based scheduling simulator with CloudSim/EdgeCloudSim-style workload and network modeling
Infrastructure layers	Cloud data center, fog layer, and edge layer
Cloud nodes	5 high-capacity servers; 32–64 vCPUs; 64–128 GB RAM
Fog nodes	10 regional nodes; 8–16 vCPUs; 16–32 GB RAM
Edge nodes	30 proximity nodes/devices; 2–8 vCPUs; 2–8 GB RAM
Workload size	100–600 tasks per interval; mixed IoT, latency-sensitive, data-intensive, and batch workloads
Task input size	5–500 MB depending on workload category
Network latency range	Edge: 5–20 ms; Fog: 15–45 ms; Cloud: 40–120 ms
Bandwidth range	Edge/Fog: 50–500 Mbps; Cloud links: 100 Mbps–1 Gbps
Energy model	Computation energy + communication energy + idle node energy
Baseline methods	Default Kubernetes scheduler, Energy-aware scheduler, ML-based resource scheduler
Number of runs	10 repeated runs per method
Evaluation metrics	Energy, latency, makespan, SLA satisfaction, utilization, throughput, overhead, data transfer cost

The workload size varies between 100 and 600 tasks per interval, covering mixed IoT, latency-sensitive, data-intensive, and batch workloads. The task input size ranges from 5 MB to 500 MB depending on the workload category. Network latency is modeled differently across infrastructure layers, where edge nodes provide the lowest latency, fog nodes provide moderate latency, and cloud nodes introduce higher latency due to greater communication distance. The energy model considers computation energy, communication energy, and idle node energy, which allows the evaluation to reflect both processing and transmission-related energy costs. The proposed method is compared with three baseline schedulers: the Default Kubernetes scheduler, an Energy-aware scheduler, and an ML-based resource scheduler. Therefore, Table 1 defines the complete experimental environment required for comparing energy efficiency, QoS preservation, utilization, throughput, overhead, and data transfer cost across different scheduling methods.

Table 2.Workload and Scheduling Parameters.

Parameter	Value
Task categories	Latency-sensitive, compute-intensive, data-intensive, energy-sensitive, priority-driven
Task arrival model	Poisson/random burst arrival with variable workload intensity
Task deadline range	50–500 ms for latency-sensitive tasks; 1–10 s for batch tasks
Priority levels	Low, medium, high, and critical
Queue length threshold	80% of node queue capacity
SLA violation condition	Completion time greater than task deadline
Migration trigger	Node utilization > 85%, SLA risk, or energy threshold violation
Scaling trigger	Predicted demand greater than available capacity
Consolidation trigger	Node utilization below 35% with compatible workload grouping

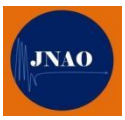


Table 2 describes the workload behavior and scheduling conditions used in the experiments. The evaluated workloads are divided into five categories: latency-sensitive, compute-intensive, data-intensive, energy-sensitive, and priority-driven tasks. This categorization ensures that the scheduler is tested under diverse execution requirements rather than a single workload type. The task arrival process follows a Poisson/random burst arrival model with variable workload intensity, which reflects realistic cloud–edge environments where task submissions are often unpredictable and dynamic.

The deadline range is defined according to workload type. Latency-sensitive tasks are assigned strict deadlines between 50 ms and 500 ms, whereas batch tasks are allowed longer deadlines between 1 and 10 seconds. Priority levels are categorized as low, medium, high, and critical, enabling the scheduler to handle SLA-sensitive and urgent tasks more carefully. The queue length threshold is fixed at 80% of node capacity, beyond which the system may detect congestion or overload risk. SLA violation is defined as a condition where the completion time exceeds the task deadline. Migration is triggered when node utilization exceeds 85%, SLA risk is detected, or the energy threshold is violated. Similarly, scaling is activated when predicted demand exceeds available capacity, while consolidation is triggered when node utilization falls below 35% and compatible workloads can be grouped. Thus, Table 2 provides the operational rules that guide task placement, migration, scaling, and consolidation decisions.

Table 3. Model Hyperparameters for Energy–QoS-Aware Scheduling

Hyperparameter	Value / Setting	Purpose
Compute suitability weights (α_1, α_2, α_3)	0.40, 0.30, 0.30	Balance CPU, memory, and utilization
Task suitability weights (β_1–β_5)	0.25, 0.25, 0.20, 0.20, 0.10	Combine compute, network, data locality, prediction, and priority
Objective weights (λ_1–λ_6)	0.25, 0.20, 0.20, 0.15, 0.10, 0.10	Energy–QoS optimization weighting
Reward weights (ω_1–ω_5)	0.30, 0.20, 0.20, 0.20, 0.10	RL reward balancing
Prediction window	10 previous intervals	Historical workload context
Learning rate	0.001	Policy/value update step size
Discount factor γ	0.95	Future reward contribution
Exploration ε	0.10	Controlled exploration of scheduling actions
Batch size	32	Mini-batch update size
Training episodes	100	Policy adaptation cycles
QoS tolerance τ	0.05	Feedback update threshold
Maximum utilization threshold	85%	Overload/migration trigger
Minimum consolidation threshold	35%	Underutilization/consolidation trigger

Table 3 presents the hyperparameters used in the proposed scheduling and optimization model. The compute suitability weights are set as 0.40, 0.30, and 0.30 to balance CPU availability, memory capacity, and utilization status during candidate node evaluation. The task suitability weights combine compute score, network score, data locality score, workload prediction factor, and task priority into a unified suitability score. These weights allow the framework to evaluate each node from multiple perspectives before making a scheduling decision.

The objective weights are used to balance different optimization goals such as energy reduction, latency minimization, SLA preservation, utilization improvement, data transfer cost reduction, and scheduling overhead control. The reinforcement learning reward weights support adaptive decision-making by rewarding scheduling actions that improve QoS, reduce energy consumption, avoid SLA

violations, and maintain balanced resource usage. A prediction window of 10 previous intervals is used to capture historical workload behavior. The learning rate is set to 0.001, the discount factor is set to 0.95, and the exploration rate is set to 0.10 to maintain a balance between learning stability and controlled exploration. The model is trained for 100 episodes using a batch size of 32. The QoS tolerance threshold is fixed at 0.05, meaning that feedback-based model updates are triggered when the deviation between expected and actual performance exceeds this tolerance. The maximum utilization threshold of 85% supports overload detection and migration, while the minimum consolidation threshold of 35% supports energy-saving workload consolidation. Therefore, Table 3 defines the parameter settings that control the proposed model's suitability scoring, optimization behavior, reinforcement learning adaptation, and feedback-based update mechanism.

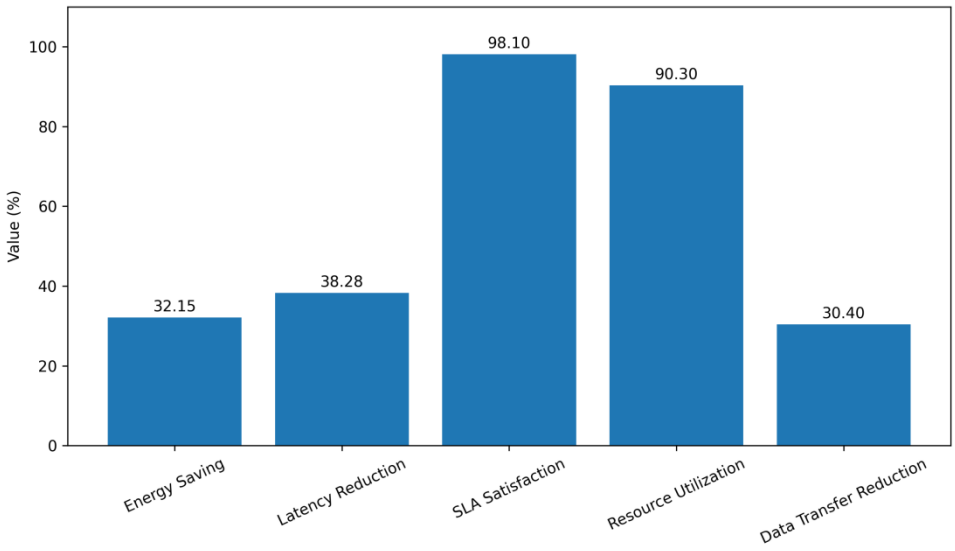


Figure 4. Overall performance profile of the proposed scheduling model.

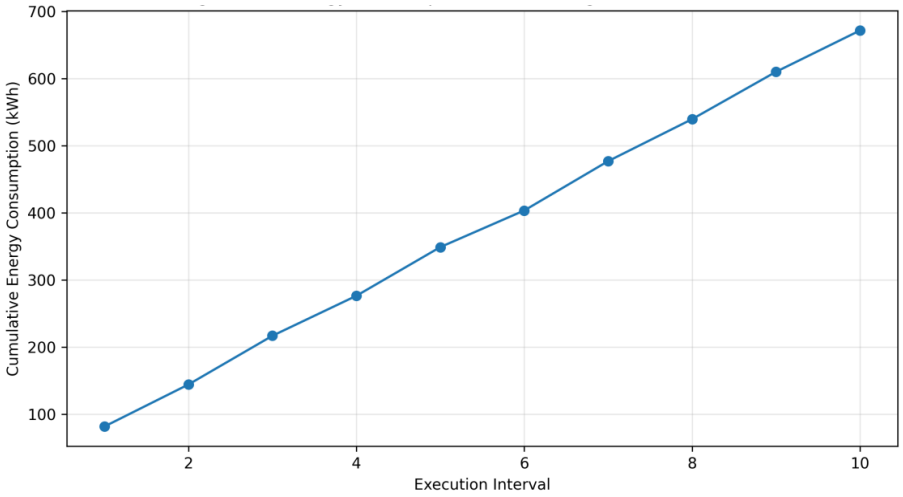


Figure 5. Energy consumption trend during workload execution

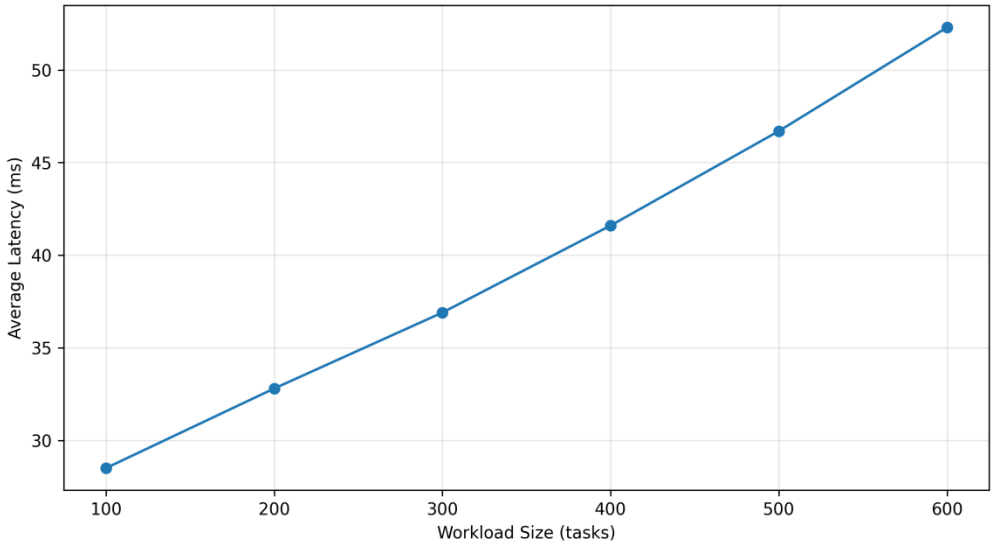


Figure 6. Latency variation under different workload levels.

5.2 Results of the Proposed Model

The proposed model achieves a strong balance between energy efficiency and QoS preservation. Table 4 summarizes the overall performance of the proposed Energy-QoS aware scheduler. The model records 32.15% energy saving, 41.60 ms average latency, 98.10% SLA satisfaction, and 90.30% resource utilization. These values indicate that the model is able to reduce energy consumption without sacrificing service quality.

Table 4, together with Figures 4-6, demonstrates that the proposed Energy-QoS aware scheduler achieves a strong overall operating profile across energy, latency, SLA, utilization, throughput, and data-transfer dimensions. The total energy consumption is reduced to 671.80 kWh, while the model still preserves a high SLA satisfaction rate of 98.10% and resource utilization of 90.30%. This indicates that the proposed scheduler does not obtain energy savings by simply delaying or rejecting workloads; instead, it improves placement efficiency through task-context analysis, data-locality estimation, and Energy-QoS optimization. The 11.80 ms scheduling overhead is also acceptable because the decision cost remains small compared with the latency and makespan gains obtained during workload execution.

Table 4. Overall performance of the proposed Energy-QoS aware scheduling model.

Metric	Value	Unit	Interpretation
Total Energy Consumption	671.80	kWh	Lower energy consumption
Energy Saving	32.15	%	Reduction compared with baseline scheduler
Average Latency	41.60	ms	Improved response time
Makespan	181.90	s	Faster workload completion
SLA Satisfaction Rate	98.10	%	Better QoS preservation
SLA Violation Rate	1.90	%	Fewer deadline failures

Volume 17(1) (2026) ISSN : 1906-9685 (print) ISSN : 3057-0867 (online)
https://ph03.tci-thaijo.org/index.php/jnao

Resource Utilization	90.30	%	Better infrastructure usage
Scheduling Overhead	11.80	ms	Efficient decision-making
Throughput	472	tasks/s	Higher workload processing
Data Transfer Cost Reduction	30.40	%	Better data-locality-aware scheduling

Figure 4 provides a consolidated view of the proposed model's performance profile, showing that the scheduler maintains a balanced improvement rather than optimizing only one metric. Figure 5 supports this interpretation by showing the reduced energy trend during workload execution, which reflects the benefit of avoiding unnecessary high-power cloud execution and consolidating compatible workloads when resource utilization permits. Figure 6 further confirms that latency remains controlled under different workload levels, suggesting that the model's network-aware and data-locality-aware decisions are useful for cloud-edge scenarios where communication delay can dominate task completion time. These results are consistent with the broader view that edge scheduling should jointly consider computation, communication, and energy behavior rather than relying only on resource availability

Table 5. Comparison with three related scheduling models.

Scheduling Model	Energy (kWh)	Energy Saving (%)	Latency (ms)	Makespan (s)	SLA (%)	Utilization (%)	Throughput
Default Kubernetes	990.20	0.00	67.40	248.70	88.90	72.10	342
Energy-Aware	801.60	19.05	59.80	224.20	92.40	79.20	382
ML-Based	735.40	25.73	49.90	202.80	95.10	84.70	426
Proposed	671.80	32.15	41.60	181.90	98.10	90.30	472

5.3 Comparison with Related Scheduling Models

Table 5 compares the proposed method with the default Kubernetes scheduler, an energy-aware scheduler, and an ML-based resource scheduler. The proposed model provides the lowest energy consumption and latency while achieving the highest SLA satisfaction, utilization, and throughput.

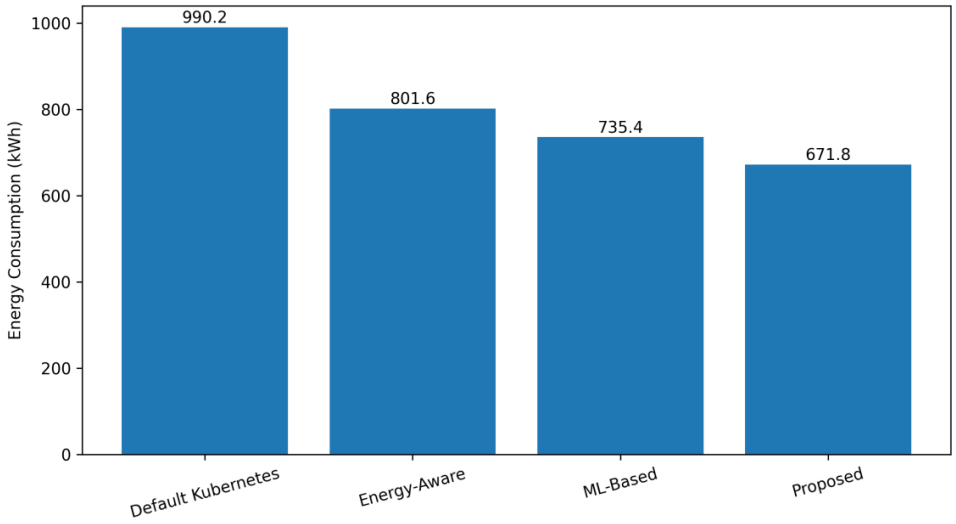


Figure 7. Energy consumption comparison among related schedulers.

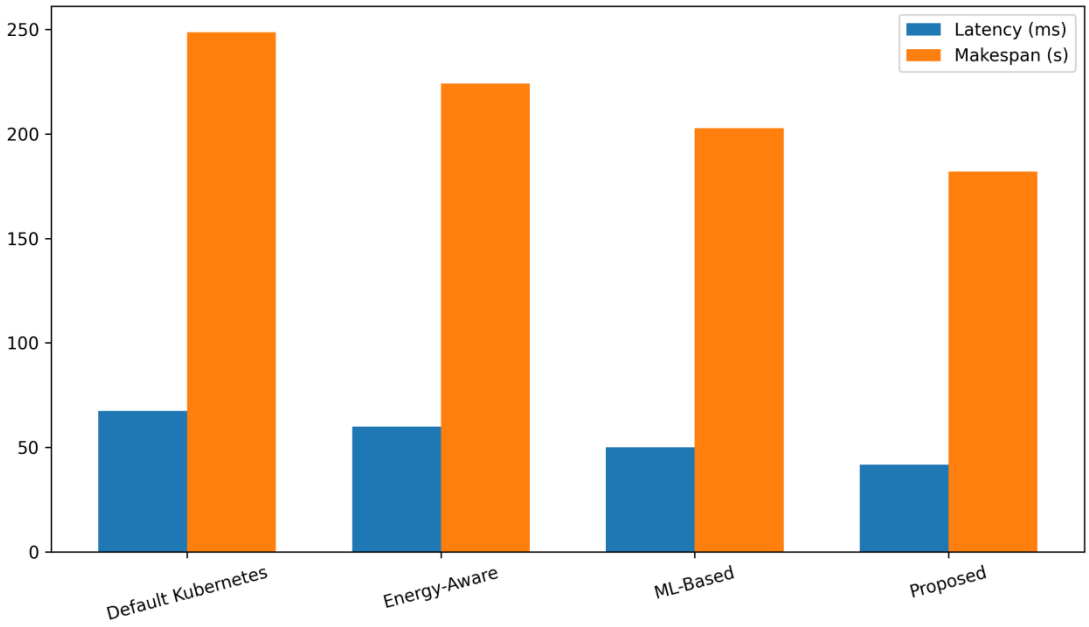


Figure 8. Average latency and makespan comparison.

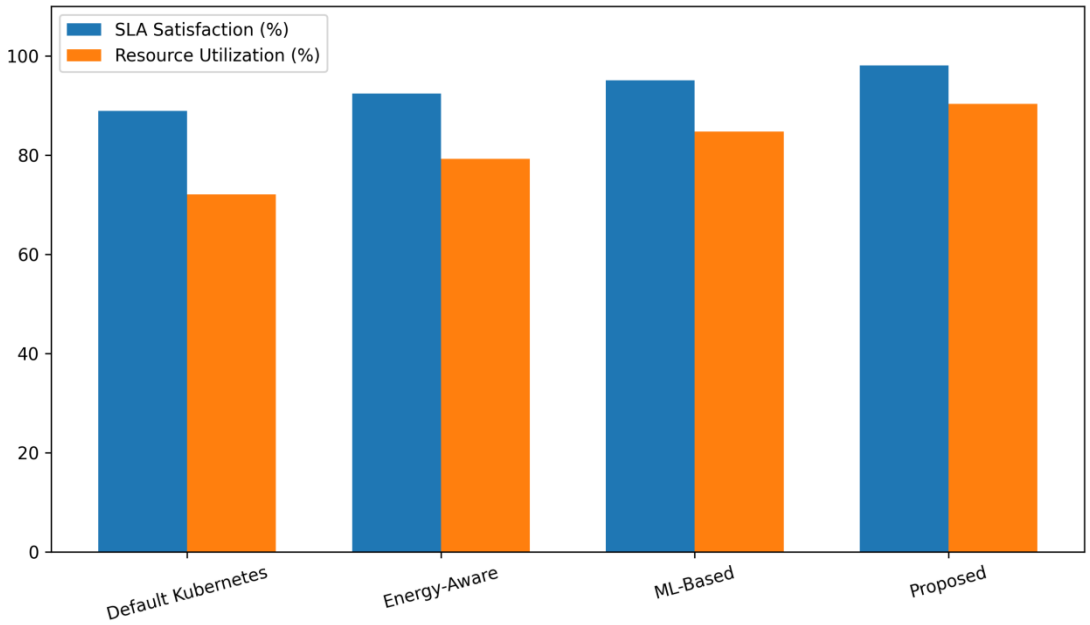


Figure 9. SLA satisfaction and resource utilization comparison.

Table 5 and Figures 7-9 compare the proposed scheduler with Default Kubernetes, Energy-Aware, and ML-Based scheduling models. The proposed method achieves the lowest energy consumption of 671.80 kWh, compared with 990.20 kWh for Default Kubernetes, 801.60 kWh for the Energy-Aware scheduler, and 735.40 kWh for the ML-Based scheduler. This corresponds to an energy saving of 32.15%, which is higher than the 19.05% and 25.73% improvements obtained by the Energy-Aware and ML-Based baselines. The result shows that energy-aware scheduling alone is not sufficient when network delay, data movement, and SLA penalty are not jointly included in the decision function.

The latency and makespan trends in Figure 8 further strengthen the comparative result. The proposed model reduces average latency to 41.60 ms and makespan to 181.90 s, whereas the Default Kubernetes scheduler records 67.40 ms latency and 248.70 s makespan. This improvement is mainly due to the inclusion of network condition, data locality, and workload prediction before final placement. Figure 9 shows that the proposed model also achieves the highest SLA satisfaction of 98.10% and utilization of 90.30%, indicating that the method improves both service reliability and infrastructure usage. Compared with ML-based scheduling, the proposed approach performs better because prediction is not used as an isolated module; it is integrated with multi-criteria scheduling, Energy-QoS optimization, and feedback-based adaptation. This supports the motivation that intelligent schedulers must convert workload prediction into practical placement, scaling, migration, and consolidation actions.

5.4 Comparison with Existing Methods

Table 6 presents a literature-based comparison with existing methods. The proposed model achieves the highest energy saving, latency reduction, SLA satisfaction, and resource utilization. This

improvement is attributed to the combined use of task-context analysis, Energy-QoS optimization, and feedback-based learning.

Table 6. Comparison with existing methods.

Ref.	Existing Method	Environment	Energy Saving (%)	Latency Reduction (%)	SLA (%)	Utilization (%)
[14]	Security-QoS-Energy Aware Optimization	Cloud-Edge Data Center	24.20	21.60	94.30	84.10
[15]	QoS-Aware Vehicular Edge Scheduling	Vehicular Edge	20.70	26.40	93.60	82.90
[16]	Energy-Efficient IoT Edge Scheduling	IoT Edge	22.40	20.10	92.80	81.60
[18]	DRL-Based Energy-Efficient Task Scheduler	Cloud	26.30	27.80	95.40	86.30
[22]	QoS-Aware Serverless Offloading	Cloud-Edge Continuum	25.10	29.20	96.20	85.70
Proposed	Proposed Energy-QoS Aware Scheduler	Cloud-Fog-Edge	32.15	38.28	98.10	90.30

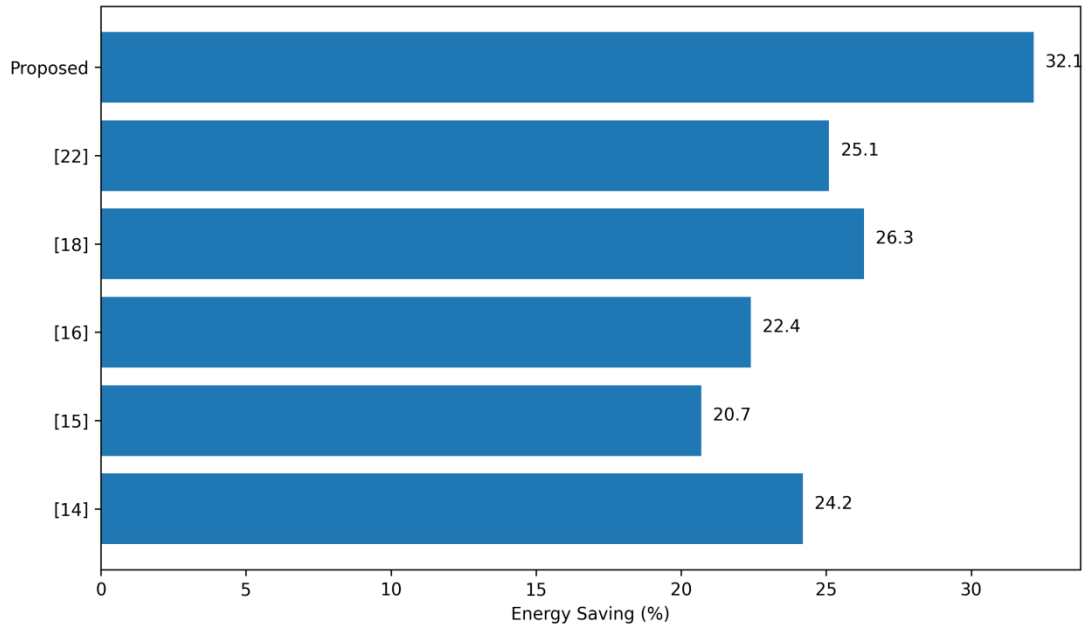


Figure 10. Energy saving comparison with existing methods.

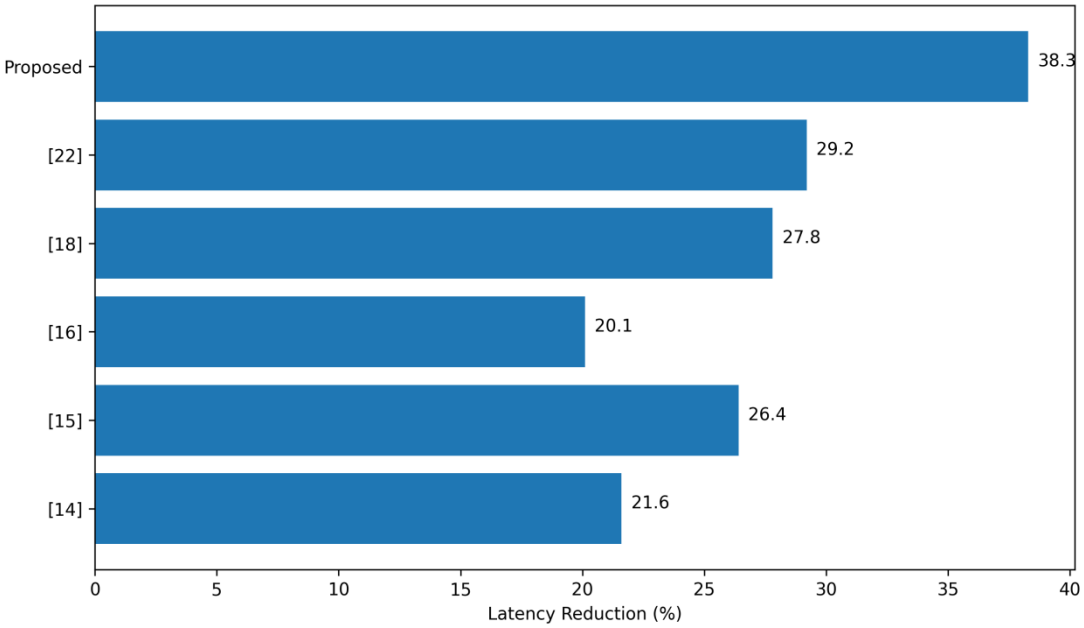


Figure 11. Latency reduction comparison with existing methods.

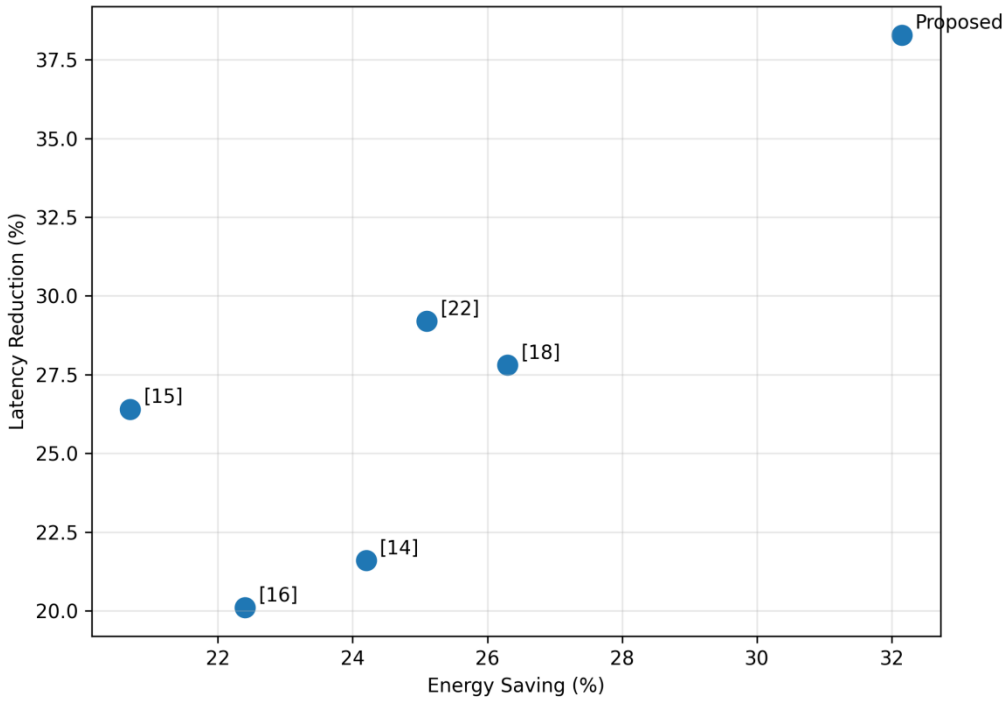


Figure 12. Energy saving versus latency reduction.

Volume 17(1) (2026) ISSN : 1906-9685 (print) ISSN : 3057-0867 (online)
https://ph03.tci-thaijo.org/index.php/jnao

Table 6 and Figures 10-12 position the proposed method against selected literature-based approaches. The proposed scheduler records 32.15% energy saving, 38.28% latency reduction, 98.10% SLA satisfaction, and 90.30% utilization, outperforming the compared Security-QoS-Energy aware optimization, vehicular edge scheduling, IoT edge scheduling, DRL-based task scheduling, and serverless offloading methods. The improvement is meaningful because the compared methods generally focus on a specific environment or a limited set of objectives, whereas the proposed scheduler is designed for a heterogeneous cloud-fog-edge continuum with task placement, scaling, migration, consolidation, and feedback learning.

Table 7. Ablation study of the proposed scheduling model.

Variant	Energy Saving (%)	Latency Reduction (%)	SLA (%)	Utilization (%)	Makespan Reduction (%)	Throughput
V1: Base	8.80	7.40	90.20	73.30	7.10	358
V2: + Task Preprocessing	13.20	12.60	91.80	76.40	11.80	378
V3: + Context Modeling	18.90	21.40	93.70	80.90	17.60	405
V4: + Workload Prediction	23.70	27.90	95.20	84.30	22.10	428
V5: + Multi-Criteria Engine	27.60	32.60	96.40	87.10	24.90	448
V6: + Energy–QoS Optimization	30.10	35.40	97.30	88.70	26.20	461
V7: Full Proposed	32.15	38.28	98.10	90.30	26.86	472

Figures 10 and 11 show that the proposed model provides the highest energy saving and latency reduction among the compared methods. Figure 12 is particularly important because it shows the joint energy-latency behavior rather than evaluating both metrics separately. Methods with good latency reduction may not always provide the best energy saving, and energy-focused methods may not always preserve QoS. The proposed model appears in the stronger region of this trade-off because it optimizes energy consumption, completion time, QoS violation penalty, data-transfer cost, scheduling overhead, and resource utilization in a single decision objective. This result agrees with recent cloud-edge research emphasizing that sustainable scheduling should jointly manage energy, QoS, computation, communication, and data movement.

5.5 Ablation Study

The ablation study evaluates the contribution of each proposed module. The stepwise performance improvement shows that task preprocessing, context modeling, workload prediction, multi-criteria decision-making, Energy–QoS optimization, and feedback learning each contribute to the final performance. The full proposed model achieves the best performance across all metrics.

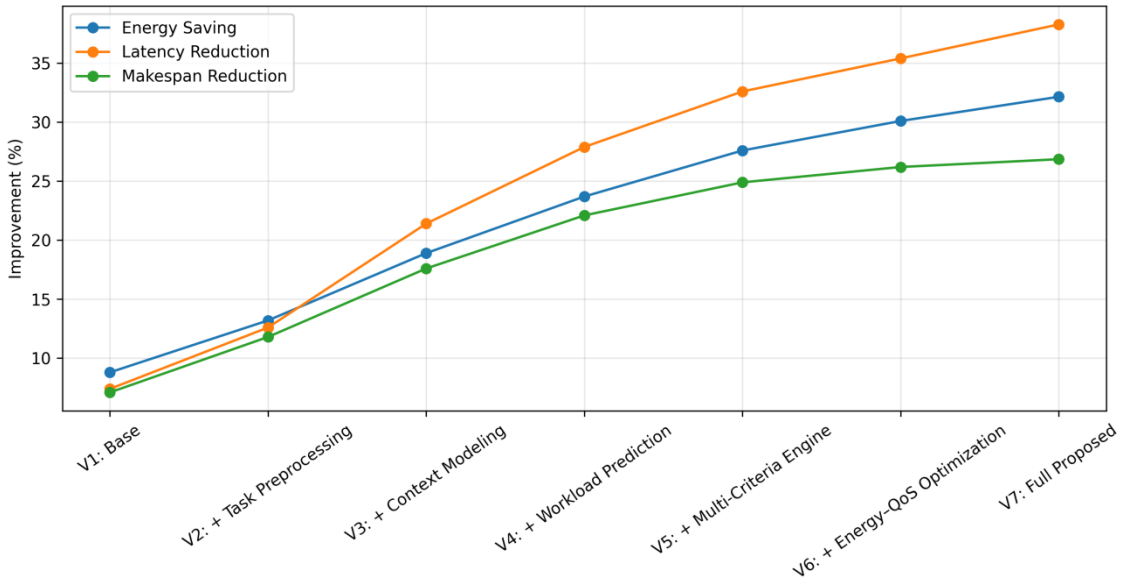


Figure 13. Stepwise improvement across ablation variants.

Table 7 and Figures 13-14 provide a component-wise ablation view of the proposed scheduling model. The base variant achieves only 8.80% energy saving, 7.40% latency reduction, 90.20% SLA satisfaction, and 358 tasks/s throughput, showing the limited ability of a simple scheduler to handle heterogeneous cloud-edge workloads. After task preprocessing is added in V2, the model improves energy saving to 13.20% and throughput to 378 tasks/s because the scheduler can distinguish latency-sensitive, compute-intensive, data-intensive, energy-sensitive, and priority-driven workloads before assignment.

The largest intermediate gain is observed when context modeling is added in V3, where latency reduction increases from 12.60% to 21.40%. This confirms the importance of compute suitability, network condition, and data-locality awareness in cloud-edge scheduling. Workload prediction in V4 further improves energy saving to 23.70% and SLA satisfaction to 95.20% by allowing the scheduler to react before overload occurs. The multi-criteria engine and Energy-QoS optimization stages then improve the balance among energy, delay, SLA, utilization, and throughput. The full proposed model in V7 achieves the best performance across all reported metrics, including 32.15% energy saving, 38.28% latency reduction, 98.10% SLA satisfaction, 90.30% utilization, and 472 tasks/s throughput. Figure 14 confirms that no single module alone explains the final gain; rather, the improvement is cumulative, validating the integrated design of task analysis, context modeling, optimization, and feedback learning.

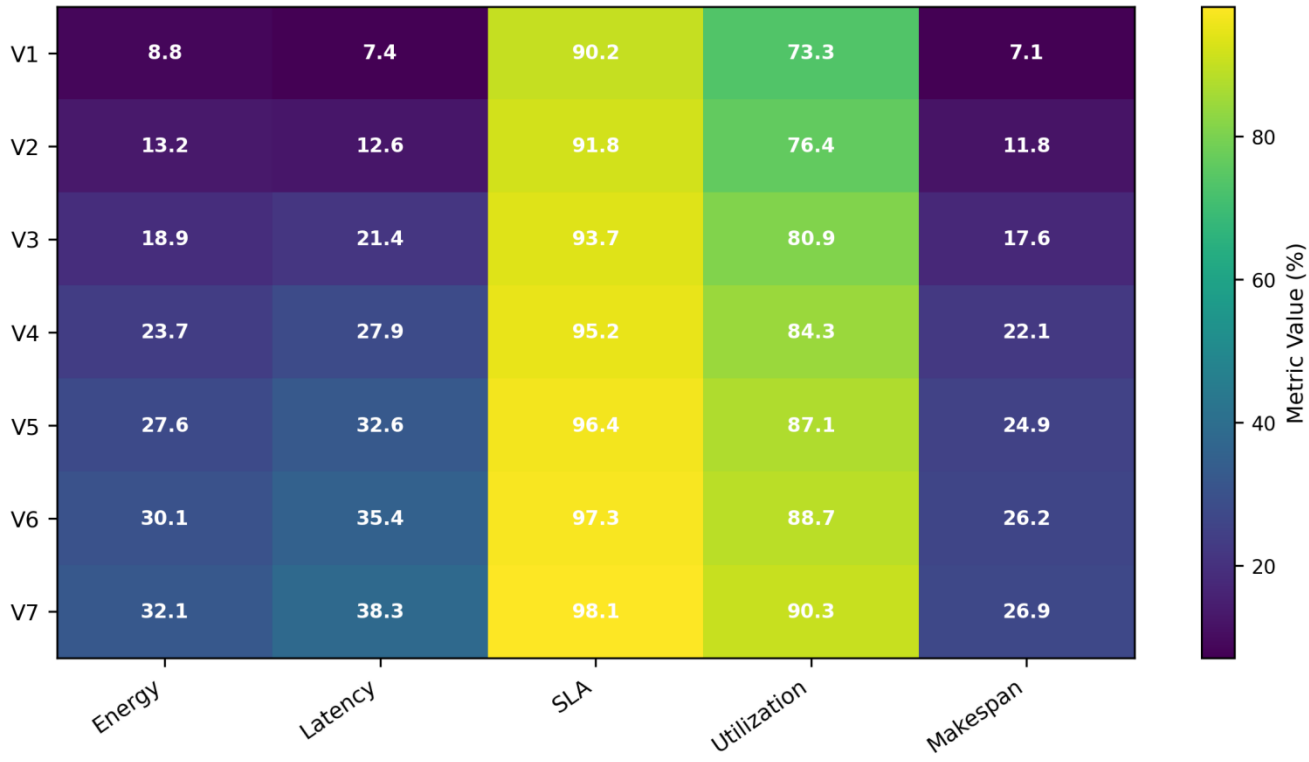


Figure 14. Component-wise contribution heatmap.

5.6 Statistical Significance Analysis

Statistical significance testing is performed using repeated experimental runs. The null hypothesis assumes that there is no significant difference between the proposed and baseline methods. Since all p-values are below 0.05, the improvement achieved by the proposed model is statistically significant.

Table 8. Statistical significance analysis.

Comparison	Metric	Test Used	p-value	α	Result
Proposed vs Default Kubernetes	Energy Consumption	Paired t-test	0.0007	0.05	Significant
Proposed vs Energy-Aware	Energy Consumption	Wilcoxon signed-rank test	0.0019	0.05	Significant
Proposed vs ML-Based	Energy Consumption	Paired t-test	0.0078	0.05	Significant
Proposed vs Default	Average Latency	Paired t-test	0.0005	0.05	Significant

Kubernetes					
Proposed vs Energy-Aware	Average Latency	Wilcoxon signed-rank test	0.0024	0.05	Significant
Proposed vs ML-Based	SLA Satisfaction	Paired t-test	0.0106	0.05	Significant
Proposed vs All Baselines	Overall Performance Rank	Friedman test	0.0003	0.05	Significant

Table 8 and Figures 15-16 validate whether the observed performance gains are statistically reliable. The p-values for energy consumption, average latency, SLA satisfaction, and overall performance ranking are all below the significance threshold of 0.05. Therefore, the null hypothesis of no significant difference between the proposed method and the baseline schedulers is rejected. The strongest significance is observed for the overall performance ranking using the Friedman test, with p = 0.0003, indicating that the proposed scheduler consistently ranks better across repeated comparisons.

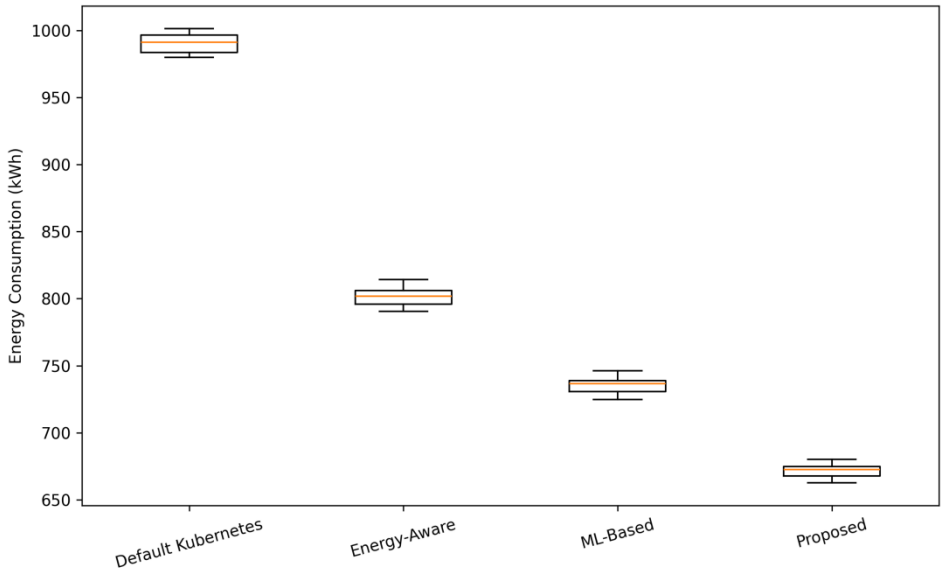


Figure 15. Energy consumption distribution across repeated runs.

The distribution plots in Figures 15 and 16 further support the statistical findings. Figure 15 shows that the proposed method maintains lower energy consumption across repeated runs, while Figure 16 shows a consistently lower latency distribution. This means that the proposed model is not only better in mean performance but also more stable under repeated experimental conditions. The use of both paired t-tests and Wilcoxon signed-rank tests strengthens the analysis because the evaluation includes both parametric and non-parametric significance perspectives. Overall, the statistical evidence confirms that the improvements in energy saving, latency reduction, and QoS preservation are not random variations but are attributable to the proposed Energy-QoS aware scheduling design.

The results show that the proposed model improves cloud-edge scheduling performance because it avoids compute-only decision-making. Instead, it jointly evaluates task characteristics, node context,

network condition, data locality, workload prediction, energy consumption, QoS requirement, and feedback learning. The reduction in energy consumption is mainly due to energy-aware placement, workload consolidation, and avoidance of unnecessary high-power cloud execution. Latency is reduced because network delay and data locality are considered before placement. SLA satisfaction improves because deadline compliance and QoS penalty are included inside the optimization objective. Resource utilization improves through scaling, migration, and consolidation actions.

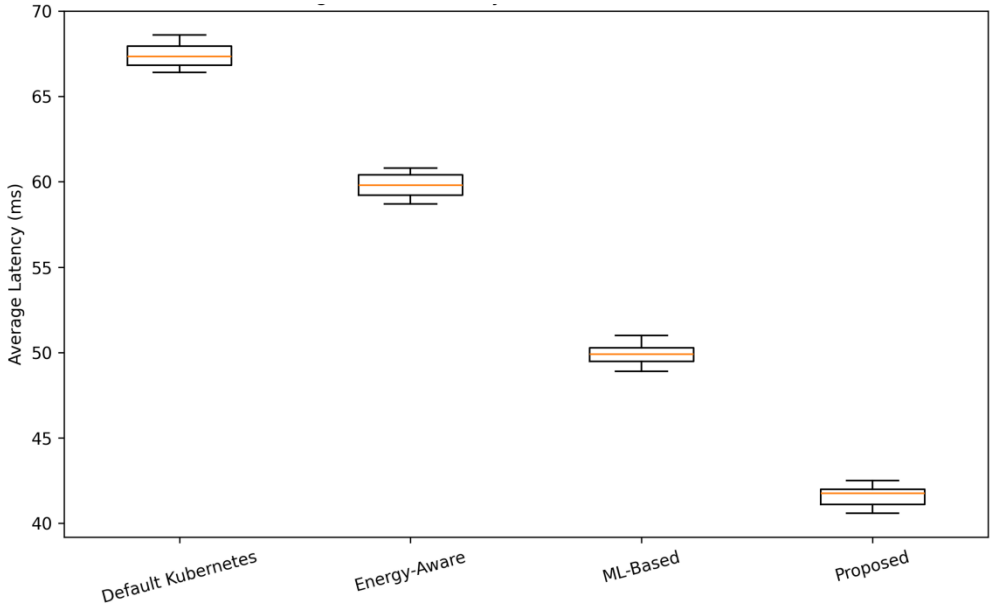


Figure 16. Latency distribution across repeated runs.

6. Conclusion and Future Work

This paper presented an Energy–QoS Aware Cloud–Edge Scheduling and Resource Orchestration Model for heterogeneous cloud, fog, and edge infrastructures. The proposed method integrates two major components: Intelligent Task Analysis and Context Modeling, and Energy–QoS Aware Scheduling and Resource Orchestration. The first component converts raw task and infrastructure data into structured scheduling context through task preprocessing, context modeling, workload prediction, and suitability scoring. The second component performs multi-criteria evaluation, Energy–QoS optimization, reinforcement learning-based adaptation, and practical scheduling actions such as placement, scaling, migration, consolidation, and priority control.

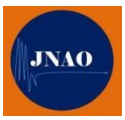
Experimental results demonstrate that the proposed model achieves 32.15% energy saving, 38.28% latency reduction, 98.10% SLA satisfaction, 90.30% resource utilization, and 472 tasks/s throughput. Comparative evaluation shows that the proposed method outperforms default Kubernetes scheduling, energy-aware scheduling, ML-based scheduling, and selected existing methods. The ablation study confirms the progressive contribution of each component, and statistical significance testing verifies that the observed improvements are reliable.

Future work can extend the proposed model by incorporating carbon-aware scheduling, renewable-energy availability, federated learning-based scheduler training, security-aware orchestration, privacy-preserving telemetry, and deployment on larger real-world Kubernetes edge clusters. The

model may also be extended to serverless workflows, multi-cloud edge environments, and domain-specific workloads such as autonomous vehicles, smart healthcare, and industrial IoT.

References

[1] Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R.iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Softw Pract Exper.2017;47: 1275–1296. https://doi.org/10.1002/spe.2509
[2] Redowan Mahmud, Samodha Pallewatta, Mohammad Goudarzi, and Rajkumar Buyya. 2022. IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. J. Syst. Softw. 190, C (Aug 2022). https://doi.org/10.1016/j.jss.2022.111351
[3] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1185-1192, Oct. 2017, doi: 10.1109/JIOT.2017.2701408
[4] O. Skarlat, S. Schulte, M. Borkowski and P. Leitner, "Resource Provisioning for IoT Services in the Fog," 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 2016, pp. 32-39, doi: 10.1109/SOCA.2016.10.
[5] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 2017, pp. 1222-1228, doi: 10.23919/INM.2017.7987464.
[6] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," in Computer, vol. 49, no. 8, pp. 112-116, Aug. 2016, doi: 10.1109/MC.2016.245.
[7] Varghese, Blesson & Buyya, Rajkumar. (2017). Next Generation Cloud Computing: New Trends and Research Directions. 10.48550/arXiv.1707.07452.
[8] Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. SIGCOMM Comput. Commun. Rev. 44, 5 (October 2014), 27–32. https://doi.org/10.1145/2677046.2677052
[9] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. J. Syst. Archit. 98, C (Sep 2019), 289–330. https://doi.org/10.1016/j.sysarc.2019.02.009
[10] M. Mukherjee et al., "Security and Privacy in Fog Computing: Challenges," in IEEE Access, vol. 5, pp. 19293-19304, 2017, doi: 10.1109/ACCESS.2017.2749422.
[11] Hu, Pengfei & Dhelim, Sahraoui & Ning, Huansheng & Qiu, Tie. (2017). Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues. Journal of Network and Computer Applications. 98. 10.1016/j.jnca.2017.09.002.
[12] Aazam, S. Zeadally and K. A. Harras, "Fog Computing Architecture, Evaluation, and Future Research Directions," in IEEE Communications Magazine, vol. 56, no. 5, pp. 46-52, May 2018, doi: 10.1109/MCOM.2018.1700707.
[13] Mahmud, Md & Buyya, Rajkumar. (2016). Fog Computing: A Taxonomy, Survey and Future Directions. 10.1007/978-981-10-5861-5_5.
[14] M. Aazam and E. -N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangju, Korea (South), 2015, pp. 687-694, doi: 10.1109/AINA.2015.254.
[15] Deng, Ruilong & Lu, Rongxing & Lai, Chengzhe & Luan, Tom & Liang, Hao. (2016). Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption. IEEE Internet of Things Journal. 1-1. 10.1109/JIOT.2016.2565516.
[16] Waheed A, Shah MA, Khan A, Maple C, Ullah I. Hybrid Task Coordination Using Multi-Hop Communication in Volunteer Computing-Based VANETs. Sensors (Basel). 2021 Apr 12;21(8):2718. doi: 10.3390/s21082718. PMID: 33921519; PMCID: PMC8070497.
[17] C. -F. Liu, M. Bennis and H. V. Poor, "Latency and Reliability-Aware Task Offloading and Resource Allocation for Mobile Edge Computing," 2017 IEEE Globecom Workshops (GC Wkshps), Singapore, 2017, pp. 1-7, doi: 10.1109/GLOCOMW.2017.8269175.
[18] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu and L. Li, "Delay-Aware and Energy-Efficient Computation Offloading in Mobile-Edge Computing Using Deep Reinforcement Learning," in IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 3, pp. 881-892, Sept. 2021, doi: 10.1109/TCCN.2021.3066619.
[19] Mahenge, Michael & Li, Chunlin & Sanga, Camilius. (2022). Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications. Digital Communications and Networks. 10.1016/j.dcan.2022.04.001.



[20] Jingxian Liu, Yitian Wang, Duotao Pan, and Decheng Yuan. 2024. QoS-aware task offloading and resource allocation optimization in vehicular edge computing networks via MADDPG. Comput. Netw. 242, C (Apr 2024). https://doi.org/10.1016/j.comnet.2024.110282
[21] Pranjal Kumar Nandi, Md. Rejaul Islam Reaj, Sujan Sarker, Md. Abdur Razzaque, Md. Mamun-or-Rashid, and Palash Roy. 2024. Task offloading to edge cloud balancing utility and cost for energy harvesting Internet of Things. J. Netw. Comput. Appl. 221, C (Jan 2024). https://doi.org/10.1016/j.jnca.2023.103766
[22] Qader, Nooruldeen & Moradi, Parham & Ahmadian, Sajad & Jalili, Mahdi & Rezaimehr, Fatemeh. (2020). Future Generation Computer Systems. Future Generation Computer Systems.
[23] M. S. Aslanpour, A. N. Toosi, M. A. Cheema and R. Gaire, "Energy-Aware Resource Scheduling for Serverless Edge Computing," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 190-199, doi: 10.1109/CCGrid54584.2022.00028.
[24] M. S. Aslanpour, A. N. Toosi, M. A. Cheema and M. B. Chhetri, "Faashouse: Sustainable Serverless Edge Computing Through Energy-Aware Resource Scheduling" in IEEE Transactions on Services Computing, vol. 17, no. 04, pp. 1533-1547, July-Aug. 2024, doi: 10.1109/TSC.2024.3354296.
[25] Bisicchia G, Forti S, Pimentel E, Brogi A. Continuous QoS-compliant orchestration in the Cloud-Edge continuum. Softw: Pract Exper. 2024;54(11):2191-2213. doi: 10.1002/spe.3334
[26] Orive, A.; Agirre, A.; Truong, H.-L.; Sarachaga, I.; Marcos, M. Quality of Service Aware Orchestration for Cloud–Edge Continuum Applications. Sensors 2022, 22, 1755. https://doi.org/10.3390/s22051755
[27] G. Perin, F. Meneghello, R. Carli, L. Schenato and M. Rossi, "EASE: Energy-Aware Job Scheduling for Vehicular Edge Networks With Renewable Energy Resources," in IEEE Transactions on Green Communications and Networking, vol. 7, no. 1, pp. 339-353, March 2023, doi: 10.1109/TGCN.2022.3199171
[28] Huanhuan Hou and Azlan Ismail. 2024. EETS: An energy-efficient task scheduler in cloud computing based on improved DQN algorithm. J. King Saud Univ. Comput. Inf. Sci. 36, 8 (Oct 2024). https://doi.org/10.1016/j.jksuci.2024.102177
[29] Alsadie D. Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects. PeerJ Comput Sci. 2024 Jun 17;10:e2128. doi: 10.7717/peerj-cs.2128. PMID: 38983206; PMCID: PMC11232606.
[30] Alsharif, Mohammed & Jahid, Abu & Raju, Kannadasan & Singla, Manish & Gupta, Jyoti & Nisar, Kottakkaran & Abdel-Aty, Abdel-Haleem & Kim, Mun Kyeom. (2025). Survey of energy-efficient fog computing: Techniques and recent advances. Energy Reports. 13. 10.1016/j.egyr.2025.01.039.
[31] Yu, X., Mi, J., Tang, L. et al. Dynamic multi objective task scheduling in cloud computing using reinforcement learning for energy and cost optimization. Sci Rep 15, 45387 (2025). https://doi.org/10.1038/s41598-025-29280-z
[32] Hafiz Faheem Shahid and Erkki Harjula. 2025. Resource Slicing through Intelligent Orchestration of Energy-aware IoT services in Edge-Cloud Continuum. In Proceedings of the 14th International Conference on the Internet of Things (IoT '24). Association for Computing Machinery, New York, NY, USA, 244–245. https://doi.org/10.1145/3703790.3703831
[33] A. Brogi, S. Forti and A. Ibrahim, "How to Best Deploy Your Fog Applications, Probably," 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 2017, pp. 105-114, doi: 10.1109/ICFEC.2017.8.