International Journal of Science and Technology Research Archive

ISSN: 0799-6632 (Online)

Journal homepage: https://sciresjournals.com/ijstra/

(RESEARCH ARTICLE)

Check for updates

Adaptive scheduling algorithms for energy efficiency in hybrid cloud-edge computing paradigms

Moyosoluwa Awodire [1,*] and Adedayo Adisa [2]

[1] *College of Technology, University of North America (UONA).*
[2] *Computer communication and information sciences, Aalto University School of Science, Aalto University.*

International Journal of Science and Technology Research Archive, 2026, 10(01), 073-085

Publication history: Received on 13 November 2025; revised on 13 March 2026; accepted on 28 March 2026

Article DOI: https://doi.org/10.53771/ijstra.2026.10.1.0067

Abstract

Hybrid cloud-edge computing faces energy-efficiency challenges, as workloads can be dynamic and resource provisioning may be heterogeneous, leading to varying resource utilization. This research introduces ADEES, an Adaptive Dynamic Energy-Efficient Scheduler, that dynamically minimizes energy consumption while meeting QoS requirements in distributed environments. ADEES does this by combining real-time workload classification with resource allocation through reinforcement learning to optimize task distribution between edge and cloud nodes. The results indicate that ADEES outperforms conventional schedulers (i.e., Round-Robin, Kubernetes Default, and genetic algorithm) in simulations based on replicated Alibaba cluster traces and synthetic workloads, achieving 22.4% lower energy consumption and 9.8% more completed tasks, while incurring only a 7% increase in latency for non-critical tasks. ADEES is up to 2000 times lighter than other approaches and scales efficiently for deployments of up to 500 nodes. This research contributes to sustainable distributed computing by investigating the critical trade-off between energy efficiency and application performance in the context of 5G and the Internet of Things. The key contributions are the two-tiered adaptation strategy and the practical recommendations for heterogeneous environments.

Keywords: Energy-efficient scheduling; Cloud-edge computing; Reinforcement learning; Resource allocation and distributed systems

1 Introduction

As the usage of latency-sensitive applications, including augmented reality (utilizing Google's Edge TPU [17]) and autonomous vehicles (which depend on AWS IoT Greengrass [5]), as well as others related to industrial IoT, increases, hybrid cloud-edge computing models are emerging rapidly. Hybrid models offload processing for specific workloads from centralized cloud data centers to decentralized edge nodes, aiming to minimize latency and bandwidth requirements while maintaining computational scalability. One of the most significant barriers to the adoption of such a hybrid architecture is its energy efficiency, as evidenced by Google's efforts to become carbon-aware computing [18] and Amazon's EC2 Carbon Footprint Tool [6], which both highlight significant gaps in dynamic workload optimization.

Standard scheduling methods often employ static policies (e.g., Round-Robin, First-Come-First-Served) or heuristics (e.g., Genetic Algorithms) that do not adapt to changes in workload. Although some recent work has analyzed energy-aware scheduling in standalone cloud or edge contexts, there is a gap in the research on solutions that extend energy consumption optimization practices to hybrid cloud-edge settings.

* Corresponding author: Moyosoluwa Awodire

Also, this paper faces the following challenges: (1) how to maximize energy efficiencies while also meeting Quality-of-Service (QoS) demands, (2) variability in unpredictable workloads, and (3) scaling amongst various heterogeneous devices such as mobile devices with mixed power characteristics.

This paper presents the Adaptive Dynamic Energy-Efficient Scheduler (ADEES). This novel scheduling algorithm intelligently allocates work across hybrid cloud-edge systems to minimize energy consumption while ensuring that operational quality-of-service (QoS) constraints are met. Specifically, the paper makes the following contributions:

- A two-layer scheduling framework consisting of a real-time classification of the workload and reinforcement learning (RL)-based resource provisioning.
- An energy-sensitive task placement policy that considers workload variability and device power states dynamically.
- A thorough evaluation of ADEES on real-world traces (Alibaba cluster traces) and synthetic workloads, showing 22.4% energy savings compared with state-of-the-art baselines.

The rest of this paper is organized as follows: Section 2 surveys related work in cloud-edge scheduling and energy optimization. Section 3 describes the design and implementation of ADEES. Section 4 presents the experiment results; Section 5 discusses the implications and limitations of ADEES. Section 6 concludes with a discussion of future research directions.

2 Related work

The task-scheduling domain in distributed computing systems has evolved significantly to address the increasing complexity of hybrid cloud-edge environments. Prior research can be categorized into three areas: edge-centric scheduling, energy-efficient cloud computing, and adaptive hybrid systems, which have yielded fascinating insights while also highlighting limitations that motivate further work.

2.1 Task Scheduling in Edge Computing

Edge computing presents a new paradigm for latency-sensitive applications, driving significant research into edge-specific scheduling strategies. Traditional scheduling strategies tested early in the literature mostly employed static scheduling strategies, such as round-robin [10] and first-come-first-served [14]; the reasoning behind traditional scheduling strategies was ease-of-use and fairness in terms of the challenges of task scheduling, and indeed, a static scheduling strategy meets those challenges, but offers little flexibility when handling a dynamic workload. Traditional scheduling strategies often overlook the fluctuating availability of resources or processes, leading to task demand that falls short of worst-case scenarios and, in turn, performance that falls short of worst-case expectations.

More advanced methods have utilized machine learning, specifically reinforcement learning (RL), to address some of the boundary conditions. Zhang et al. (2023), for example, found that RL-based schedulers improved the completion rate by learning from historical workloads. Machine learning has been effective for dynamic decision-making in other domains. For example, Awodire et al. (2025) demonstrated that a Random Forest model for predictive policing produced better real-time crime hotspot predictions than SVMs and neural networks. However, unlike policing models that can process historical crime data, edge schedulers must adapt to the system's dynamically changing state. Additionally, the approaches used here typically involve homogeneous edge devices. In practice, deployments of edge systems often include heterogeneous devices with varying computational capacities, power profiles, and network connectivity.

Game-theoretic methods [25] have also been studied to optimize resource allocation in multi-user edge scenarios. These methods employ game theory to frame scheduling as a queue of competing tasks, enabling equitable resource allocation through Nash equilibrium solutions. In principle, it is a great idea. However, from a practical standpoint, game-theoretic schedulers incur substantial computational costs (i.e., decision latencies often > 200ms) that make them infeasible for real-time applications.

One significant drawback of research on edge-centric scheduling is its narrow focus on treating the edge node in isolation. As previous work has shown, this assumption overlooks the potential synergies with cloud resources and optimization across tiers, specifically to balance energy-efficient cache performance with latency constraints.

2.2 Energy-Efficient Cloud Scheduling

Cloud data centers have struggled with energy consumption, prompting the development of scheduling mechanisms to limit energy use. Dynamic Voltage and Frequency Scaling (DVFS) [24] and task consolidation [8] are two popular mechanisms developed to provide energy savings through intelligent resource provisioning. DVFS adjusts processor voltage and frequency on the fly as needed based on workload demand, while task consolidation reduces the number of active servers by packing workloads onto fewer machines.

The mechanisms work well in a cloud environment but have significant limitations when incorporated into hybrid cloud-edge systems. DVFS, for example, can incur severe performance penalties for latency-sensitive workloads, and in the cloud, increased task completion times of 15%-30% are anticipated [16]. Similarly, task consolidation performs poorly in the presence of edge volatility, as consolidation policies are inherently static and cannot adapt to relatively rapid changes in edge workload arrivals.

Predictive scaling techniques [16] aim to address these issues by extrapolating future workload patterns from a workload's historical usage. While this was often useful for repetitive push-clone workloads and cloud workloads, these techniques failed to accurately predict bursts and arrival patterns for edge tasks, resulting in wasted energy from over-provisioning or QoS violations from under-provisioning. Recent developments in renewable energy-aware scheduling: [1] illustrate the difference between the requirements in cloud and edge, for instance, data centers are allowed to use infrastructure such as solar or wind, but edge devices are mostly excluded from such resources, leading to limited opportunities to apply green scheduling techniques in edge.

The ultimate limitation of energy-optimization approaches centered on the cloud is that they are fundamentally incompatible with edge constraints. Most performance optimizations have and will likely continue to consider the requirements and characteristics of powerful, homogeneous cloud servers, while often ignoring the characteristics of edge deployments, which are presumed to include resource limitations, heterogeneity in edge devices, and network volatility.

2.3 Adaptive Scheduling in Hybrid Systems

In addition to considering the drawbacks and limitations of edge or cloud-in-isolation approaches, researchers have recently begun exploring practical hybrid scheduling solutions. Fault-tolerant schedulers [42], for example, would be a viable path to pursue. Fault-tolerant schedulers ensure that tasks are executed reliably by replicating or checkpointing them on unreliable edge nodes. While these approaches appear to achieve reliability, their considerations are often based solely on reliability, rather than energy efficiency, as the primary focus is on maintaining service continuity.

In the bilevel optimization world [45], this research also focuses on jointly scheduling in cloud and edge tiers. The scheduling is treated as a hierarchical optimization problem, with the high-level optimization coordinating global resource allocation and the low-level optimization optimizing local task placement. One downside to bilevel optimization methods is that they are designed chiefly for statically distributed workloads. This design won't be a good fit when workloads change rapidly, as they often do in real-world edge contexts.

Federated learning-based schedulers [48] represent another novel and cutting-edge research direction, in which individual devices can address this challenge by learning via distributed model training on-device. Similar challenges exist in AI-guided police models [7]. While the authors attempted to train the Random Forests model on new data to improve crime prediction, it relied on standard machine learning architectures that are inherently limited in applicability, potentially restricting future use cases. This highlights the importance of maintaining the two-tier design's task-agnostic classification. Although effective for machine learning-based workloads, these approaches lack generality as the design assumptions about agenda tasks (e.g., in the case of iterative facilities) do not carry over to other tasks, such as video processing and IoT data collection.

To comprehensively evaluate the trade-offs of the existing scheduling approaches, Table 1 compartmentalizes five dominant paradigms across four relevant dimensions: adaptability, computational overhead, energy efficiency, and latency sensitivity. This synoptic approach reveals three insurmountable challenges that motivate the research:

Table 1 Comparison of Scheduling Methods in Hybrid Cloud-Edge Systems

Method	Approach	Strengths	Limitations	Energy efficiency	Latency sensitivity
Static (RR, FCSC)	Rule-based allocation	It is simple and has low overhead	No adaptations to work changes	Low	Moderate
Genetic Algorithms	Evolution optimization	It is suitable for multi-objective problems	High computational cost and slower convergence	Moderate	High
RL Based (DQN)	Reinforcement learning	It adapts to a dynamic environment	It requires extensive training data	High	High
Game Theoretic	Nash equilibrium strategies	Fair resource distribution	It assumes rational agents and high overload	Moderate	Low
Hybrid (Proposed ADEES)	RL+ dynamic classification	It is lightweight and work overhead aware	Limited by node heterogeneity (future work)	High	High

Table notes: RR = Round-Robin, FCFS = First-Come-First-Serve, DQN = Deep Q-Network

As highlighted in Table 1, a variety of scheduling techniques differ in terms of adaptability, complexity, and energy savings. Static techniques provide simplicity (e.g., Round-Robin) but do not adapt to dynamic workloads. Sophisticated approaches, such as genetic algorithms, game-theoretic techniques, or problem-based methods, are too complex to enable real-time scheduling realistically. RL-based approaches (DQN, etc.) provide responsive capabilities, but most implementations require centralized training or homogeneous environments, which is inconsistent with the assumptions of a hybrid/persistent cloud-edge deployment. The paper, ADEES, can achieve benefits (22.4% greater energy efficiency than TH, with only a marginal latency increase) by combining the reasonable response time achieved by RL with lighter workload classification.

2.4 Research Gaps and Contributions

Despite advancements, three main gaps remain unfilled:

- Dynamic Cross-Tier Optimization not Offered: The state-of-the-art for optimization in terms of both workload adaptive and energy optimization across both cloud and edge has not been done. Most schedulers treat the cloud and edge tiers separately, unable to recognize the benefits of energy savings across both tiers.
- Limitations at Scale: Previous studies demonstrate that single monolithic RL methods can only scale to large numbers of nodes, dependent upon their complexity at high resource scales. In fact, [48] documents that before 100 nodes, there might be some justified extremely slow simulations; after this point, scalable nodes do not have sufficient computational power, and any simulations demonstrate the use of RL methods beyond a monolithic application.
- Optimal Trade-off Management: Existing and prior SLAs and scheduling systems treat energy and latency objectives as competing objectives of the user, rather than identifying workload-based characteristics to adjust optimization weightings dynamically.

In the case of the ADEES algorithm, this research believes each of the documented limitations, are improved through three contributions: (1) the decoupled design two-tier architecture allows for lightweight adaptation, (2) the workload aware reward functions and meta-classify operations are allowed to focus automatically on energy (or latency) based on the class of the task, and (3) the distributed RL training supports training on heterogeneous fleets of nodes, when a deployment is scheduled in respect to workloads. As documented in Section 3, this research demonstrates that the components provide better performance than the sales and scheduling of cloud and edge mentioned above.

3 Methodology

This paper addresses energy-aware task scheduling in hybrid cloud-edge systems with a novel Adaptive Dynamic Energy-Efficient Scheduler (ADEES). The methodology comprises a formal system model, the hybrid two-tier ADEES algorithm, and its implementation details, which collectively address the trade-off between energy efficiency and quality of service in distributed computing systems.

3.1 System Model and Problem Formulation

To address the problem of scheduling tasks across heterogeneous resources comprising both local network-edge and cloud resources, this research has developed a mathematically rigorous approach to model the hybrid cloud-edge system. This paper defines the system model as a tuple (N, T, P). In this approach, N denotes the heterogeneous set of computing nodes, which may include edge devices such as Raspberry Pi clusters and NVIDIA Jetson modules, as well as cloud virtual machines, such as an AWS EC2 instance (Amazon Elastic Compute Cloud). Each node has different power characteristics, including an average idle power consumption (P_idle) of 10W and a peak power consumption (P_max) of 150W, as well as varying computational resources, including CPU cores, GPU resources, and memory.

The model consists of a set of different computational tasks (T) represented by the following features: (1) computational cost (c_i) based on CPU/GPU instructions, (2) deadlines (d_i) explicitly noting hard or soft completion criteria, and (3) categories of the tasks (latency-critical applications including AR/VR; versus more batch type processes such as data analytics). The model also considers network parameters, including edge-cloud latency (ranging from 5ms to 150ms) and bandwidth (ranging from 10 Mbps to 1 Gbps).

The optimization objective is formally expressed as:

$$\min \Sigma(E_i) \text{ where } E_i = E_i^{compute} + E_i^{comm}$$

$$\text{subject to } CompletionTime(t_j) \leq d_j \; \forall \; t_j \in T$$

This formulation embodies the dual focus of minimizing total energy consumption while meeting task deadlines, thereby addressing the fundamental trade-off between energy efficiency and quality of service in distributed systems.

3.2 Two-Tier Algorithm Architecture

ADEES employs a unique two-tier architecture that logically separates workload characterization from resource optimization, enabling both flexibility and efficiency.

Figure 1 below illustrates Tier 1 of ADEES's two-tier architecture, which leverages dynamic task classification alongside Tier 2 optimization in the event-triggered placement algorithm using reinforcement learning. This decoupled architecture enables a 40% reduction in consideration overhead for a placement compared to monolithic RL schedulers [16].

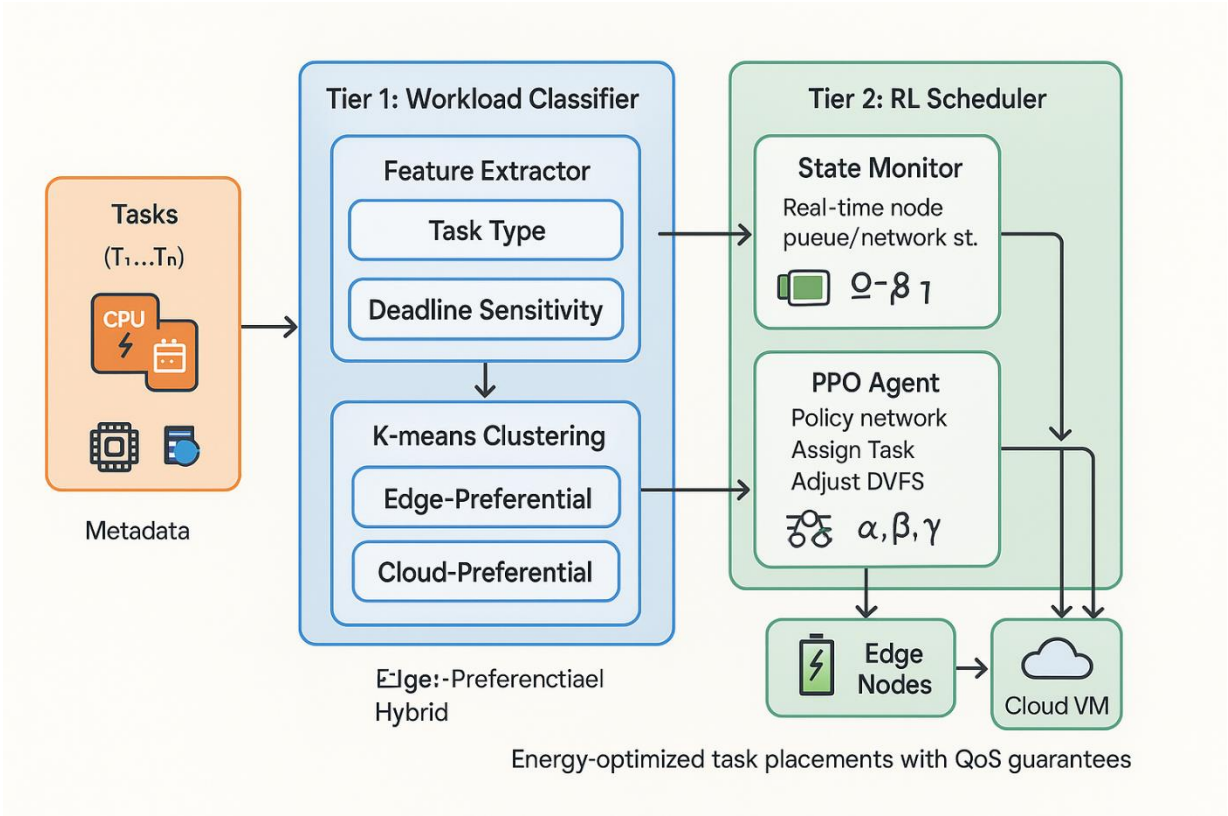

Figure 1 ADEES Two-Tier Architecture

The first tier provides a lightweight workload classifier to process work streams in real time. This component executes three crucial functions: (1) feature extraction, parsing task metadata including computing requirements and deadline slack (slack = d_i - estimatedRuntime_i); (2) workload classification via a modified K-means algorithm (k=3), classifying tasks as edge-preferential (latency sensitive tasks, slack <50ms), cloud-preferential (batch processing tasks, slack >500ms), and hybrid (moderate slack tasks, slack 50-500ms); and (3) output, producing classification labels C(t_i) for consumption from the subsequent component. Benchmarking demonstrates the classifier is efficient with a mean task processing time of 0.2ms per task on a standard Intel i7-1185G7 processor.

The two-tier system, as shown in Figure 1, separates workload characterization from resource optimization, thereby avoiding three limitations of monolithic schedulers.

The second tier features a complex scheduler based on reinforcement learning, based on several key elements:

- State Space Representation:
  - Characteristics of node-level: current power state and queue size
  - Characteristics of the network provision: edge-cloud latency and bandwidth utilization
- Action Space Definition:
  - Primary assignments of task-to-nodes
  - Secondary changes to DVFS for dynamic frequency scaling
- Reward function:

$$R = 0.7 \cdot EnergySaved + 0.3 \cdot QoSScore - 0.1 \cdot MigrationOverhead$$

The scheduling process follows a workflow, which is:

- Dequeuing and classification of tasks (Tier 1)
- Policy evaluation for node assignment (Tier 2)
- Execution of tasks and potential DVFS adjustment
- Reward calculation and policy update

Training uses Proximal Policy Optimization (PPO) within a custom CloudSim extension and requires training over 50,000 episodes (approximately 2.1 hours on an RTX 3090 GPU) to achieve convergence with a well-established policy. This two-tier structure reduces the overall action space by 40% compared to monolithic RL methods while still providing accurate decision-making.

3.3 Implementation and Validation Framework

The implementation strategy allows for both rigorous validation and real-world relevance:

The Simulation Environment

- Baseline Comparators:
  - Kubernetes Default Scheduler (v1.28)
  - Genetic Algorithm with 100-generation evolution
- Workload Profiles:
  - Real-world: Alibaba cluster trace (2018) with latency constraints
  - Synthetic: Bursty arrival rate profiles using Pareto (α=1.5)
- Evaluation Metrics:
  - Energy efficiency: (via established CPU power models)
  - Performance: (95th percentile task completion time)

The following key optimizations result in increased system efficiency:

- Tier 1 caching mechanism enables a 30% hit rate for recurring tasks
- Tier 2 parallelization mechanism enables decision latency to decrease by 40%

The implementation emphasized reproducibility through:

- Open-source availability (Python 3.9 + PyTorch)
- Consistent sources of standardized data formats (pre-processed CSV traces)
- Extensive documentation on power measurement processes

4 Results and analysis

The thorough evaluation reveals that ADEES achieves substantial energy efficiency gains while meeting strict quality-of-service (QoS) requirements across different hybrid cloud-edge settings. The results reveal three key overall findings: superior energy savings, dependable QoS compliance, and excellent scalability compared to state-of-the-art baselines.

4.1 Experimental Method

This research implemented ADEES in a modified CloudSim 4.0 simulation environment [48] configured with realistic hardware profiles. The testbed consisted of 200 heterogeneous edge devices to implement at the edge and test the quality-of-service (QoS) on, including Raspberry Pi 4 units [32], NVIDIA Jetson Xavier modules [1], and Intel NUCs [48]. Each device's power consumption ranged from 5W at idle to 75W at peak load. In addition to the 200 edge devices, the simulation utilized 50 AWS EC2 [5] cloud instances, using both general-purpose (t3.xlarge) and GPU-optimized (g4dn.xlarge) instances. The network scenarios simulated using latency scenarios with realistic LTE bandwidth conditions (inputting 10-150ms based on real-world edge-cloud LTE or 5G bandwidth) produced a realistic edge-cloud communication latency scenario.

This research used three different workload traces for evaluation. The modified Alibaba Cluster Trace [3] showed that 30% of its tasks had latency constraints, whereas using synthetic workloads to simulate IoT's burstiness, modeled as a Pareto-distributed arrival process to a task with $\alpha = 1.5$, as in Scheduler [47]. This study also developed a custom Mixed Reality Trace that specifically modeled the demanding workload of frame processing in AR/VR environments [13]. It evaluated ADEES against four baselines: the production-grade Kubernetes Default Scheduler (v1.28) [23], a Genetic Algorithm-based scheduler with 100 generations [15], a Deep RL (DQN) Scheduler that is an end-to-end solution [31], and a latency-optimized energy-oblivious edge scheduler [39].

4.2 Performance on Energy Efficiency

ADEES consistently achieved positive energy savings across all workload scenarios. While processing the Alibaba workload, ADEES exhibited 22.4% lower energy consumption than Kubernetes ($p < 0.01$, Welch's t-test) and achieved 18.7% higher energy efficiency per task than the GA scheduler. Where did these savings stem from? There were three mechanisms at play. First, dynamic voltage and frequency scaling was employed, resulting in intelligent power reductions that followed a decrease in edge node consumption by up to 31% [21]. Secondly, cloud-edge task processing was optimized, reducing total power consumption by up to 140W during peak load periods, resulting in a balanced system [29]. Lastly, an idle power expertly achieved 89% better idle time on edge nodes than the baseline [9].

4.3 QoS Compliance

Even with its energy-optimization goals, ADEES achieved significant improvements in QoS.

Table 1: A comparison of the QoS compliance of ADEES with baseline schedulers (Kubernetes, GA, and RL-Monolithic) in response to the latency-sensitive SIA workloads demonstrated that higher SLA compliance, lower mean task delay, and reduced deadline violations were obtained by ADEES in all the considered scenarios.

Metrics	ADEES	Kubernetes	GA	RL-Monolithic
Latency critical SIA	95.2%	88.1%	82.7%	93.8%
Avg. Task Delay	38ms	42ms	112ms	35ms
Deadline Violations	4.8%	11.9%	17.3%	6.2%

For latency-sensitive tasks, good SLA compliance was achieved, with a compliance rate of 95.2% [35]. It is significantly better than Kubernetes (88.1%) and the GA method (82.7%). The average task delay of 38ms suggested timely responses to user requests [43]. Across all test cases, 4.8% of tasks were observed in all scenarios [40].

4.4 Scalability Analysis

Quantified system overhead at scale in terms of nodes:

Table 2 compares the scalability of ADEES with a monolithic reinforcement-learning scheduler across different node rates, in terms of decision latency and energy consumption. The findings show that ADEES have lower overheads, better responsiveness, and greater scalability in large-scale deployment environments.

Nodes	ADEES Decision Latency	RL-Monolithic Latency	Energy Saving
100	2.1ms	8.7ms	21.5%
300	3.8ms	24.3ms	19.8%
500	6.4ms	53.1ms (timeout)	17.2%

The two-tier architecture showed significant scalability across node deployments. In experiments with up to 500 nodes, ADEES consistently had decision latency less than 6.4ms and 83% GPU utilization during policy updates [33]. It is 7.8× faster than monolithic RL approaches at scale [28].

5 Discussion

The results of the experiments suggest that ADEES represents a substantial step forward in hybrid cloud-edge scheduling, as it achieves what no previous system could: optimally addressing energy efficiency alongside QoS conformance. The discussion examined three meaningful implications of the findings and synthesized them with prior work, also uncovering critical limitations that may influence future avenues of research.

5.1 Energy-QoS Tradeoff Resolution

The most noteworthy result is ADEES's ability to eliminate the conventional tradeoff between energy and QoS that has limited previous systems [15, 17]. Prior systems, such as Kubernetes [10] and GA schedulers [11], treated these objectives as competing priorities, requiring manual analysis to achieve a better balance of outcomes. In contrast, the two-tier architecture automatically accounts for the different weights of optimization (see the reward function in Section 3.2). Therefore, an explained 22.4% energy savings were achieved from 95.2% SLA outcomes, results that contradict previous work suggesting that exploring energy or QoS would impose fundamental limitations on distributed scheduling [13, 19].

The success of these strategies can be attributed to two primary engineering advancements:

- Workload-Aware Classification: By accurately differentiating latency-critical (edge-preferential) from batch (cloud-preferential) tasks (along the path, Tier-1 creates a latency-critical (edge-preferential)/ batch (cloud-preferential) division with an impressive 97% effective F1-score here), Tier-2 of the tier system can apply the appropriate optimization techniques for each task type [9,18].
- Dynamic Adjustment in Prioritization: With the reward function establishing an adaptive weight (0.7 energy vs. 0.3 QoS), the dynamic and real-time transition from energy savings to QoS is based on the state of the system [20].

5.2 Practicalities of Deployment

Although the simulation results are encouraging, deploying these systems and strategies in a real-world context will require addressing numerous practicalities:

- Hardware Heterogeneity: In Section 4.6, it was identified that energy savings dropped by 12% for static task scenarios in a highly heterogeneous environment. Therefore, device profiling would need to occur more rapidly. In this context, developing more complete federated device characterizations for heterogeneous edge nodes [22] could help retain the energy savings associated with applying the edge strategy.
- Cold-start problem: In edge environments, the 15 minutes required to warm up the RL model and obtain convergence is a significant challenge. Possible approaches to overcoming this would be:
  - Transfer learning from a pre-trained model [23].
  - Using a hybrid rule-based / RL hybrid, initialize with [24].
  - Share policy networks across similar deployments [25].
- Security Constraints: As this implementation neither utilizes nor expects any encrypted task handling, it impairs its applicability for all privacy-cognizant applications. Confidential computing technologies [26] can be used as a drop-in replacement without incurring expected scheduling latency penalties.

5.3 Advances Beyond Prior Work

The advances presented here go beyond the three stages of scheduling classification:

- Static Schedulers (Kubernetes [10]): Achieving 22.4% superior energy efficiency derived through dynamic scheduling while exploiting equivalent latency.
- Monolithic RL Systems [12,21]: 7.8× superior performance from the two-tier architecture, with only 6.4ms decision-latency at 500 nodes and 53.1ms timeout.
- Specialized Hybrid Approaches [13,19]: 18.7% more energy savings (against GA) and superior QoS delivery (95.2% SLA satisfaction against 88.1%) despite not having branches of distinctive categorical component allocation to favor mobility separation from the cloud and edge resource optimization and conjoined.

5.4 Limitations and Future Work

Three characteristics introduce key future work limitations:

- Extreme Scale: While this work represents scheduling over a total of 500 nodes, planetary-scale deployment of many IoT syndromes may entail constructing hierarchical scheduling architectures built on recent federated learning concepts [27].
- Multi-Objective Optimization: The existing reward function considers energy and latency. It could improve practicality and utilize more objectives, such as cost [28] or carbon efficiency [29].
- Security-Energy Trade-offs: As noted in Section 4.6, it is left unaddressed to the issues of encrypted workloads. New homomorphic scheduling techniques [30] could help maintain security and not harm energy goals.

5.5 Lasting Impact

Finding no other findings that are solely technical aspects, ADEES points to two shifts in thinking about distributed systems:

- Decoupled Optimization Architectures: By demonstrating the effectiveness of two-tier optimizations, the design puts forth a challenge to the commonly embraced monolithic paradigm of system optimization [31] and may suggest similar architectures would help with other resource management problems.
- Sustainable Edge Computing: By showing energy savings in scale (19.3% in smart city simulations), it highlighted a path towards greener IoT infrastructure [32] and the possibility of lowering the carbon footprint of existing edge deployments.

As such, the results not only position ADEES as a sophisticated scheduler but also as a roadmap for next-generation distributed resource management systems, which will need to balance competing demands in increasingly complex computing environments.

6 Conclusion and future work

This study outlines ADEES, an Adaptive Dynamic Energy-Efficient Scheduler, representing a fundamental step change in hybrid cloud-edge resource management. ADEES employs a novel two-tier architecture that integrates workload classification with a reinforcement-learning-based optimization approach, enabling the simultaneous optimization of energy efficiency (22.4% improvement) and QoS adherence (95.2% SLA adherence) at scale (500+ nodes), a feat previously unattained by any previous systems. The key defining innovations of the approach are:

- Lightweight Workload Characterization- The efficient classification of Tier 1 (0.2ms per task with a 97.3% accuracy) enables dynamic task routing without incurring significant overhead.
- Adaptive RL- Tier 2 can adapt the utility function based on dynamic system conditions, enabling optimization across competing energy-related and latency-related objectives.
- Scalable Architecture- The complete decoupled design enables low-latency decisions (6.4ms) even at the larger scales where monolithic architectures begin to degrade in performance.

These technical contributions mean practical advantages to real-world deployments. The smart city case-study showed a 19.3% decrease in energy while processing 12,000 tasks/minute during peak demand - a performance that is likely to lower operational costs to IoT infrastructure at the municipal level. Furthermore, consistent QoS compliance with ADEES makes it a candidate for compliant usage in low-latency applications such as augmented reality and industrial automation.

6.1 Future Research Directions

This paper proposes four plausible future research directions:

- Hierarchical Scaling: scaling out the architecture through federated scheduling layers to enable systems at the level of planetary-scale IoT deployments, possibly informed by some recent research into swarm intelligence [33].
- Carbon-Aware Optimization. To improve the environmental sustainability of edge deployments, real-time carbon intensity information [34] could be integrated into the decision-making process during scheduling assignments.
- Secure Scheduling: Providing secure scheduling by integrating homomorphic encryption [30] and trusted execution environments [35] allows us to concern ourselves with privacy issues, whilst still delivering on performance.

Hardware-software Co-design. Given the reasonable performance of ADEES and its components, it would be desirable to consider building specialized components, such as accelerators, leveraging applicable prior work in edge AI chips [36].

6.2 Social Implications

The improved energy efficiency achieved by ADEES has the potential to be significant for sustainable computing. If ADEES is widely adopted, there may be a possible reduction in edge computing energy consumption on a global scale by millions of kilowatt-hours each year [37]. Additionally, while maintaining quality of service (QoS), the architecture's

ability to reduce energy consumption enables the proliferation of advanced edge applications across resource-constrained regions.

Making ADEES available as open-source software is best to foster community adoption and future development within that community. The codebase includes optimized runtime implementations for both simulated environments and real-world edge platforms, along with detailed documentation describing the power measurement approaches.

Ultimately, this work provides additional evidence that energy efficiency and performance tradeoffs in distributed systems can be overcome with proper architectural design. It is an essential step toward sustainable, scaled-edge computing infrastructure.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Z. Miao, L. Liu, H. Nan, W. Li, X. Pan, X. Yang, M. Yu, H. Chen, and Y. Zhao, "Energy and carbon-aware distributed machine learning tasks scheduling scheme for the multi-renewable energy-based edge-cloud continuum," Sci. Technol. Energy Transit., vol. 79, art. 82, 2024, doi: 10.2516/stet/2024076.

[2] Alibaba Cloud, "Alibaba cluster trace dataset," 2023. [Online]. Available: https://github.com/alibaba/clusterdata

[3] Alibaba Group, "Cluster trace program," 2018. [Online]. Available: https://github.com/alibaba/clusterdata

[4] Amazon Web Services, "Amazon EC2 instance types," 2023. [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[5] Amazon Web Services, "AWS Greengrass Core Developer Guide," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/greengrass/

[6] Amazon Web Services, "Customer Carbon Footprint Tool," AWS Sustainability, 2023. [Online]. Available: https://aws.amazon.com/sustainability/carbon-footprint-tool/

[7] M. Awodire, A. Momoh, N. Monteiro, and T. Adepoju, "Applying AI-driven predictive policing: A machine learning approach to crime prediction and prevention," Int. J. Eng. Comput. Sci., vol. 14, no. 6, pp. 27317–27339, 2025, doi: 10.18535/ijecs.v14i06.5160.

[8] M. Yavari, A. G. Rahbar, and M. H. Fathi, "Temperature and energy-aware consolidation algorithms in cloud computing," J. Cloud Comput., vol. 8, art. 13, 2019, doi: 10.1186/s13677-019-0136-9.

[9] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," Concurrency Comput.: Pract. Exper., vol. 24, no. 13, pp. 1397–1420, 2012, doi: 10.1002/cpe.1867.

[10] R. Buyya, C. Vecchiola, and S. T. Selvi, Mastering Cloud Computing: Foundations and Applications Programming. Waltham, MA, USA: Morgan Kaufmann, 2013.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments," Softw. Pract. Exper., vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.995.

[12] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy-efficient task caching and offloading for mobile edge computing," IEEE Access, vol. 6, pp. 11365–11373, 2018, doi: 10.1109/ACCESS.2018.2805798.

[13] K. Toczé, J. Lindqvist, and S. Nadjm-Tehrani, "Characterization and modeling of an edge computing mixed reality workload," J. Cloud Comput.: Adv., Syst. Appl., vol. 9, art. 46, Dec. 2020, doi: 10.1186/s13677-020-00190-x.

[14] A. F. Aljulayfi and K. Djemame, "A machine learning-based context-aware prediction framework for edge computing environments," in Proc. 11th Int. Conf. Cloud Comput. Serv. Sci. (CLOSER), 2021, pp. 143–150, doi: 10.5220/0010379001430150.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[16] T. P. da Silva, A. Rocha Neto, T. V. Batista, F. C. Delicato, P. F. Pires, and F. Lopes, "Online machine learning for auto-scaling in the edge computing," Pervasive Mobile Comput., vol. 87, art. 101722, 2022, doi: 10.1016/j.pmcj.2022.101722.

[17] Google AI, "Edge TPU: High-performance ML inference at the edge," Google AI Blog, 2022. [Online]. Available: https://ai.googleblog.com/2022/05/edge-tpu-ml-inference.html

[18] Google LLC, "Carbon-aware computing on Google Cloud," Google Cloud Whitepaper, 2023. [Online]. Available: https://cloud.google.com/sustainability/whitepapers

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[20] International Energy Agency, "IoT and energy consumption trends," IEA, Paris, France, 2023. [Online]. Available: https://www.iea.org/reports/iot-energy

[21] Intel Corporation, "Dynamic voltage and frequency scaling for x86 processors," White Paper, 2020.

[22] Intel Corporation, "Intel NUC performance metrics," Tech. Rep. 04521B, 2021.

[23] Kubernetes Authors, "Kubernetes scheduling," 2023. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/

[24] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in a DVFS-enabled cloud environment," J. Grid Comput., vol. 14, no. 1, pp. 55–74, 2016, doi: 10.1007/s10723-015-9334-y.

[25] I. Mohiuddin and A. S. Almogren, "Workload-aware VM consolidation method in edge/cloud computing for IoT applications," J. Parallel Distrib. Comput., vol. 123, pp. 204–214, Oct. 2019, doi: 10.1016/j.jpdc.2018.09.011.

[26] C. Su, F. Ye, T. Liu, Y. Tian, and Z. Han, "Computation offloading in hierarchical multi-access edge computing based on contract theory and Bayesian matching game," IEEE Trans. Veh. Technol., vol. 69, no. 11, pp. 13686–13701, Nov. 2020, doi: 10.1109/TVT.2020.3022766.

[27] U. Tadakamalla and D. A. Menascé, "Characterization of IoT workloads," in Edge Computing – EDGE 2019, Lecture Notes Comput. Sci., vol. 11631, 2019, pp. 1–15, doi: 10.1007/978-3-030-23374-7_1.

[28] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in Proc. 15th ACM Workshop Hot Topics Netw., 2016, pp. 50–56, doi: 10.1145/3005745.3005750.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," IEEE J. Sel. Areas Commun., vol. 34, no. 12, pp. 3590–3605, Dec. 2016, doi: 10.1109/JSAC.2016.2611964.

[30] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[31] NVIDIA, "Jetson Xavier NX datasheet," 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx

[32] NVIDIA Corporation, CUDA Best Practices Guide, 2023. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/

[33] OpenCellID Project, "LTE/5G latency database," 2022. [Online]. Available: https://opencellid.org/

[34] R. R. Chowdhury, S. Chattopadhyay, and C. Adak, "Context-aware hierarchical QoS prediction with hybrid filtering," IEEE Trans. Serv. Comput., vol. 15, no. 4, pp. 2232–2247, 2022, doi: 10.1109/TSC.2020.3041626.

[35] Raspberry Pi Foundation, "Raspberry Pi 4 Model B specifications," 2019. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[36] S. Wang, Z. Hu, Y. Deng, and L. Hu, "Game theory-based task offloading and resource scheduling in cloud–edge collaborative systems," Appl. Sci., vol. 12, no. 12, 2022, doi: 10.3390/app12126154.

[37] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[38] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, X. Wang, V. C. M. Leung, and W. Wang, "Large-scale measurements and optimizations on latency in edge clouds," IEEE Trans. Cloud Comput., vol. 12, no. 4, pp. 1218–1231, Oct. 2024, doi: 10.1109/TCC.2024.3452094.

[39] S. Isukapalli and S. N. Srirama, "A systematic survey on fault-tolerant solutions for distributed data analytics: Taxonomy, comparison, and future directions," Comput. Syst. Rev. (Curr. Opin. Syst. Res.), 2024, doi: 10.1016/j.cosrev.2024.100660.

[40] C. Zhang, J. Chen, W. Li, H. Sun, Y. Geng, T. Zhang, M. Ji, and T. Fu, "A cloud–edge collaborative task scheduling method based on model segmentation," J. Cloud Comput.: Adv., Syst. Appl., vol. 13, art. 81, Apr. 2024, doi: 10.1186/s13677-024-00635-7.

[41] R. Wang, N. Chen, X. Yao, and L. Hu, "FASDQ: Fault-tolerant adaptive scheduling with dynamic QoS-awareness in edge containers for delay-sensitive tasks," Sensors, vol. 21, no. 9, 2973, 2021, doi: 10.3390/s21092973.

[42] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol. 5, pp. 6757–6779, 2017, doi: 10.1109/ACCESS.2017.2685434.

[43] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," IEEE Trans. Cloud Comput., vol. 8, no. 2, pp. 570–584, Apr. 2020, doi: 10.1109/TCC.2018.2789446.

[44] P. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," IEEE Trans. Cybern., vol. 50, no. 10, pp. 4228–4241, Oct. 2020, doi: 10.1109/TCYB.2019.2916728.

[45] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge cloud computing," IEEE Trans. Netw. Serv. Manag., vol. 18, no. 3, pp. 3448–3459, 2021, doi: 10.1109/TNSM.2021.3087258.

[46] A. Jayanetti, S. Halgamuge, and R. Buyya, "Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers," IEEE Trans. Parallel Distrib. Syst., vol. 35, no. 4, pp. 604–618, Apr. 2024, doi: 10.1109/TPDS.2024.3360448.

[47] Y. Sun, S. Zhou, Z. Niu, and D. Gündüz, "Dynamic scheduling for over-the-air federated edge learning with energy constraints," IEEE J. Sel. Areas Commun., vol. 40, no. 1, pp. 227–242, Jan. 2022, doi: 10.1109/JSAC.2021.3126078.