Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

AI-Driven Multi-Objective Scheduling and Load Balancing of Containers in Federated Cloud Environments

Votte Rajashekhar

raj.votte@gmail.com

Malla Reddy Engineering College for Women

N. Radhika

Geethanjali college of Engineering and Technology

G Naga Rama Devi

Sreyas Institute of Engineering and Technology

Yedida Subrahmanyam

Malla Reddy Technical Campus (A Constituent Unit of Malla Reddy Vishwavidyapeeth, Deemed to be University)

N. Rahul Pal

Aditya University

Gadde Mamatha

Koneru Lakshmaiah Education Foundation

Research Article

Keywords: Federated Cloud Scheduling, Container Load Balancing, Multi-Objective Optimization, Graph Neural Networks, Kubernetes Orchestration

Posted Date: July 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10458990/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

AI-Driven Multi-Objective Scheduling and Load Balancing of Containers in Federated Cloud Environments

Votte Rajashekhar[1*], N.Radhika[2], G Naga Rama Devi[3], Yedida Subrahmanyam[4], N.Rahul Pal[5], Gadde Mamatha[6]

[1]Assistant Professor, Department of Computer Science and Engineering, Malla Reddy Engineering College for Women, Secunderabad, Hyderabad, Telangana, India
raj.votte@gmail.com

[2]Assistant Professor, Department C Science (Cyber Security), Geethanjali college of Engineering and Technology, Cheeryal, Medchal District, Hyderabad, Telangana, India.

radhikackv29@gmail.com

[3]Department of CSE, Sreyas Institute of Engineering and Technology, Hyderabad, Telangana, India.

gnagaramadevi@gmail.com

[4]Assistant Professor, Department of Computer Science and Engineering, Malla Reddy Technical Campus (A Constituent Unit of Malla Reddy Vishwavidyapeeth, Deemed to be University), Hyderabad, India.

subrahmanyamyedida@gmail.com

[5]Assistant Professor, Department of AIML, Aditya University, Aditya Nagar, Surampalem 533437, Kakinada District, Andhra Pradesh, India.

rahulpaln@adityauniversity.in

[6]Assistant Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Hyderabad, Bowtampet, Telangana -500043, India.

g.mamatha@klh.edu.in

[*]Corresponding Authors

Abstract

Federated cloud environments are gaining momentum for hosting latency-sensitive, geo-distributed, containerised applications, yet efficient scheduling in such environments remains a significant challenge. Current approaches use heuristic, rule-based, or single-objective techniques that prioritise latency, cost, or energy performance in isolation, thereby complicating SLO adherence, inter-cluster traffic, fairness, and migration stability. These limitations make them less suitable for more dynamic multi-region cloud environments with high variability and heterogeneities in workloads and resources. Motivated by this gap, this paper introduces an AI-based framework that performs multi-objective scheduling and load balancing in federated cloud environments. It incorporates a graph neural network (GNN)-based tail-latency predictor; energy and cost estimators; an NSGA-II-based Pareto optimiser; a migration-aware feasibility guard; policy distillation; and a contextual bandit for adaptive online policy selection. Our algorithm minimises SLO violations, energy consumption, operational cost, and inter-cluster traffic while maximising fairness and jointly optimising stability. Specifically, experimental evaluation based on Google Cluster Workload Traces and DeathStarBench microservice benchmarks shows that the proposed scheduler achieves the lowest SLO violation rate of 3.28%, the lowest energy consumption of 132.94 kWh, the lowest operational cost of $389.27, and the lowest inter-cluster traffic of 248.36 GB. Also achieved lower p95 and p99 latencies of 148.2 ms and 196.4 ms, respectively, 6 hotspot incidents, and a fairness index of 0.93. The results validate that the presented framework

is a pragmatic, adaptive, and scalable solution for intelligent container-based federated cloud scheduling, with high utility for Next Generation Edge–Cloud Orchestration and Distributed Service Management.

Keywords - Federated Cloud Scheduling, Container Load Balancing, Multi-Objective Optimization, Graph Neural Networks, Kubernetes Orchestration

1. INTRODUCTION

Huge advances in the seamless orchestration of applications across federated cloud environments have laid the necessary backbone of modern distributed computing to meet the demands of latency-sensitive applications, geographically distributed services, and large-scale containerised workloads spanning multiple cloud and edge regions. Deployment and scaling have become considerably easier in such settings due to container orchestration platforms like Kubernetes, but scheduling across federated clusters of containers is still a hard problem. In contrast to single-cluster settings, federated cloud scheduling needs to handle cross-region heterogeneity (of compute capacity, network latency, power characteristics, and economic cost) without violating scheduling service-level objectives (SLOs). Due to the need to meet several requirements, scheduling is inherently a multi-objective problem, where optimisations in one dimension (e.g., minimising latency) might come at the expense of another (e.g., energy consumption or inter-cluster traffic). Hence, traditional resource allocation techniques are typically ill-suited to the requirements of modern multi-region cloud systems.

Hope you will enjoy the read. There are several approaches to container scheduling and load balancing in cloud and edge–cloud settings in recent literature. Related Work Existing work has optimised Kubernetes for scheduling [1] and multi-cluster networking [2]. At the same time, other studies have examined edge container orchestration [3], energy-aware workload placement [4], and cost-sensitive federated resource management [5]. Previous work focuses on latency-aware microservice placement [1], adaptive orchestration mechanisms [2] and AI-assisted orchestration strategies that can achieve individual operational objectives. [3] Nonetheless, all these approaches are still constrained by one or more limitations. In a significant fraction of the current state of the art, the problem is still addressed using heuristics or static rules that focus on optimising a single target, e.g., delay, cost, or resource densification for energy savings [6], [7]. Despite the use of machine learning, it is almost always limited to independent prediction tasks and rarely used as part of a comprehensive end-to-end scheduling pipeline [8]. As a result, existing methods fail to adequately account for the joint effects of workload dynamics, cross-cluster latency fluctuations, migration instability, fairness, and online adaptability in a federated cloud setting.

In view of this gap, we propose GOLTI. This novel scheduling framework jointly optimises several conflicting objectives by modelling the underlying tuning dynamics and continuously constraining the search space within which the systems are conditionally optimal. The current research complements this by introducing a federated cloud environment with an AI framework for multi-objective scheduling and load balancing. At a high level, this work aims to do two things: 1) develop an adaptive scheduler that achieves minimal SLO violations, energy, operational cost, and inter-cluster traffic, while maximising fairness and placement stability. Specifically, the study combines predictive modelling, evolutionary optimisation, and online learning into a unified decision-making framework designed to operate under realistic workload and network variability.

In this work, we propose a federated scheduling architecture that combines several complementary mechanisms, the novelty of which lies in how they are integrated. We begin by proposing a GNN-based predictor to infer tail-latency patterns from microservice dependency graphs and network-aware placement constraints. Second, instead of fixed-priority heuristics, a Pareto-based NSGA-II optimisation engine is used to yield non-dominated scheduling decisions across multiple objectives. Third, we add a migration-aware feasibility guard to mitigate oscillation placement patterns and superfluous inter-cluster migrations. Fourth, to enable rapid online adaptation, a policy distillation module and a contextual bandit selector are used to improve policy choices as workloads change over time iteratively. When combined, these factors constitute a scheduling framework that is predictive, optimisation-driven, and operates in real time.

This paper makes four contributions to research. First, it encapsulates federated container scheduling as a novel, unified multi-objective optimisation problem that integrates service quality, energy, cost, communication

overhead, and fairness within a single mathematical framework. Second, it introduces a hybrid AI-based scheduling architecture that leverages low-latency prediction, objective estimation, Pareto front approximation, migration control, and online policy learning to support more robust decision-making. Third, over 30 evaluation results (using public workload traces and microservice benchmarks) show that it outperforms representative baselines in tail latency, SLO compliance, energy efficiency and cost-effectiveness, traffic shaping, fairness, and hotspot mitigation. Finally, it provides an ablation study and a lifelong learning analysis that elucidate the individual contributions of the proposed modules and confirm the framework's ability to adapt to changing operating conditions. All of these contributions, combined, push the state of the art in federated cloud orchestration beyond independent scheduling objectives toward a more realistic, adaptive, and scalable approach.

The rest of the paper is structured as follows. The second section contains a literature review on container scheduling, federated cloud orchestration, multi-objective resource allocation, and AI-driven scheduling approaches. Preliminaries are covered in Section 3, where we provide the background concepts, mathematical formulation, and foundations for system modelling to put the problem definition in perspective. The proposed method is outlined in Section 4, which includes the architectural framework, datasets, scheduler design, evaluation framework, and algorithmic description. In Section 5, we present the experimental results through comparative analysis, ablation studies, adaptation evaluation, and computational scalability analysis. Conclusions and study limitations are reported in section 6. Lastly, Section 7 concludes the paper and suggests possible directions for future research to extend intelligent scheduling in federated cloud environments.

2. RELATED WORK

Container scheduling, orchestration, and load balancing have been widely investigated in federated cloud environments due to their ability to provide scalable, latency-aware, and resource-aware management of distributed applications. Existing studies include multi-cluster orchestration, multi-objective optimisation, fog-edge-cloud resource management, DRL-based scheduling, and trust-aware secure migration. Here we summarise such contributions by theme.

2.1 AI-Based Container Scheduling and Orchestration in Federated Clouds

Container orchestration has progressed from static, often rule-based systems to modern, intelligent container orchestration frameworks powered by AI and ML that automate deployment, scaling, monitoring, and management in heterogeneous and federated cloud environments. Many studies emphasise the importance of optimised placement strategies in multi-cluster and hybrid cloud–edge settings. According to Tabatabaei [1] and Tabatabaei [25], CNF placement in distributed clusters can achieve more than 40% latency savings if the placement solution accounts for resource availability, inter-cluster traffic, and access costs. Networking overhead affecting scheduling decisions in multi-cluster Kubernetes has been demonstrated by Malleni [2], who compared two approaches to inter-cluster connectivity (in this work, we focus on Submariner and Cilium). Abbas [3] details the broader orchestration challenges across edge and fog ecosystems, identifying ongoing gaps in lightweight orchestration, cold-start mitigation, and mobility-aware offloading, whilst Caderno [4] supports the assertion that opportunistic scheduling can drastically improve performance for big-data workloads.

This calls for elastic, context-aware orchestration solutions for cloud–edge continuity. According to Ramesh [5], DevOps-integrated scaling pipelines also need to be relevant, enabling cloud-native MLOps frameworks to deliver faster deployment cycles and adaptive scheduling. Michaelis [6] adds latency probes to service meshes to improve routing and load balancing, thereby reducing tail latency in multi-cluster deployments. Microservice Placement with Mobility-Aware Operations for IoT Applications [7]. This paper discusses issues such as edge congestion and dynamic user trajectories, showing that they must be considered when designing scheduling heuristics. At the same time, decentralised orchestration models [8], such as those offered by Sofia, reduce dependence on a centralised Kubernetes control plane and provide high availability and resilience to partial failures through gossip-based coordination.

Orchestration complexity is influenced by automated device onboarding in IoT-fed federated clouds. Mafeni [9] reduces deployment overhead by semi-automating registration, and [10] proposes a distributed attribute-based access control model for placement-policy-compliant allocation of VMs for cross-cluster cloud bursting. A new dimension is introduced with semantics-driven orchestration: ontology-driven swarm orchestration, proposed by

Anuraj [11], to synchronise semantics among microservices at the device level. Şenel [12] describes work on the evolution of container isolation and CaaS multitenancy in hybrid environments, providing secure container sandboxes for untrusted workloads.

A broader orchestration imperative is introduced by the cloud-to-things continuum [12], as captured by Ullah [13] and Sofia [14], who advocate cross-layer orchestration pipelines integrating compute, network, and context telemetry. Rejiba [15] surveyed research and industry work on Kubernetes scheduling, while Tran [16] surveyed fault-tolerant Kubernetes systems; both further provided a rationale for telemetry-driven adaptive and predictive orchestration. At the intent-modelling and portability layer, Tomarchio [57] proposes a TOSCA-based orchestrator called Torch, which can mask provider-specific heterogeneity from the orchestration logic implementation, and automatically deploy, scale and rollback containerised multi-cloud applications, which highlights the requirement of TOSCA-based declarative and provider-independent specifications of container scheduling in the hybrid cloud container set-up.

2.2 Multi-Objective Optimisation and Load Balancing Approaches

Federal cloud architectures that optimise multiple conflicting performance metrics jointly, such as latency, energy, cost, security, and quality of service, have multi-objective scheduling and load balancing at their core. As shown by Taleb [17], optimal routing and application placement for XR workloads are highly non-trivial, as ultra-low-latency objectives often conflict with bandwidth and compute constraints, necessitating sophisticated routing and scheduling policies. Exploiting the rise of AI-guided auto-tuning in cloud deployments, deep learning-based solutions for resource optimisation, such as Deep-Hill [18], leverage transformer models to analyse historical scaling actions by an application and predict optimal configurations.

Resource slicing in 6G introduces novel multi-objective constraints, and the authors in [19] point out that federated scheduling should be applied in large-scale, heterogeneous cloud–fog ecosystems. An AI-based multi-access scheduling architecture like REASON, proposed by Katsaros et al. [20], naturally fuses intent-oriented decisions of resource allocation for terrestrial, satellite, and Wi-Fi links. DRL has also been used, as demonstrated by Subhan [21], to analyse various approaches to federated learning-based multi-objective QoE optimisation to enhance rich-media delivery in sliced networks.

Going beyond terrestrial networks, Iqbal [56] considers the impacts of AI-based design techniques on non-terrestrial networks, such as satellites and UAVs, demonstrating how the additional spatial and mobility facets that are enabled by deep reinforcement learning (DRL) for example, in the context of beam management and trajectory optimization will require additional dimensions of consideration in the future cloud–NTN ecosystems and thus dictating that multi-objective schedulers in such environments will need to jointly deal with the trade-offs associated with cost, latency, and reliability.

These studies on fog computing scheduling and resource management are related and help shed more light on multi-objective optimisation. Alsadie [22] presents a survey of AI-based resource management principles, citing MARL and digital twins as broader solutions for addressing conflicting quality of service criteria. Somantri [23] investigates green cloud architectures and shows emissions reduction using renewable-aware scheduling, while Rezazadeh [24] shows SHAP enables easy explainability for radio resource allocations pointing to a growing demand for transparent multi-objective schedulers.

For example, some probabilistic load balancing models (e.g., ProbLB by Sefati [26]) combine LSTM forecasts with Bayesian optimisation to deliver predictions and adaptively distribute workloads in dynamic multi-cloud environments, thereby minimising latency and SLA violations. Incorporating sustainability as a first-class objective is also reinforced by the benefits of carbon-aware scheduling frameworks such as GreenTrans [27].

Besides, Boubaker [29] implies that the ability to make trade-offs among throughput, latency, fairness, and energy consumption when scheduling multiple objects at scale over Fog-IoT resources should relate to Multi-Objective scheduling frameworks, while Vetriveeran [28], focusing on fog provisioning, surveys even demonstrated this in large-scale IoT resource management. Syed [30] also performs AIaaS architectural analyses and finds that the multi-objective constraints interact with each other when deployed across cloud–edge continua for ML inference.

2.3 Fog–Edge–Cloud Continuum Resource Management

Such an edge–fog–cloud continuum imposes specific constraints, including those arising from mobility, intermittent connectivity, device heterogeneity, and delay sensitivity. It is shown in [31] that model-order reduction enhances the efficiency of AI-based scheduling by reducing the DRL state spaces, making them suitable for resource-constrained edge environments. Qiu [32] uses data graph dependencies to make scheduling decisions and improve batch placement, while Hady [33] reviews the scope of MARL as an optimisation for distributed resources.

López Fernández [48] reviews biclustering techniques for big data and HPC platforms, with special attention to scalability and data-movement bottlenecks (e.g., data locality), which are directly relevant to the placement of containerised, data-intensive workloads across federated clouds, where minimising cross-site communication overhead is desirable.

Chia Chuan [51] provides a focused review of fog computing scheduling, concluding that hybrid heuristic–AI strategies are most effective under mobility, multi-tenancy, and real-time constraints, thereby motivating continuum-aware schedulers that jointly reason over fog, edge, and core resources in federated environments.

Placement strategies have also been inspired by privacy-preserving cloud models. Federated learning-based distributed cloud systems: (in)corporating privacy-preserving scheduling in user-level priority clouds with strict compliance requirements, Rahdari [34]. On the other hand, Donatti [35] highlights the growing trend of implementing energy-aware allocation in network slicing for container deployment across a set of federated edge clouds.

Reward-shaping methods, such as AutoShape by Chahoud [36], relevant to fog and federated cloud scheduling improve DRL convergence in such dynamic resource environments. Besides, they enable functions such as distributed tomography (Akdemir [37]) or microservices lifecycle optimisation (Moreschini [38]), highlighting the need for scalable scheduling heuristics that must be deployed across tightly resource-limited environments in Edge AI applications.

Federated systems based on blockchain introduce even more limitations. In the same vein, Roopa [39] explores the synergy architectures of blockchain–AI for wireless networks, and Arvind [40] demonstrates that an IoT-enabled warehouse system requires load-aware scheduling to ensure on-time analytics. For example, Jiang [41] on distributed data pipelines and Kabir [42] on intelligent cyber-physical systems have both highlighted that resource-aware service orchestration at the edge is central to real-time performance.

Knowledge-Graph-based Scheduling for Multi-Layer Networks (Ghosh [43]), Cross-Domain Orchestration Frameworks (Ghosh [43]), and QoX-Aware Load Balancing (Abdalzaher [44]) work on data-quality-centric IoT systems. The same limitations are reflected in the artistic analyses of adaptive streaming [45] as well as in learned lessons from the micro-edge platform for industrial IoT [46], both of which leverage time-sensitive constraints while orchestrating containers.

The digital twin-aided scheduling proposed by Aghazadeh [47], Sheraz [54], and Kamdjou [55] highlights the importance of predictive synchronisation and resource slicing, which demonstrate some degree of autonomy; thus, these approaches are relevant for federated cloud systems, where higher levels of autonomy are necessary.

2.4 Deep Reinforcement Learning for Distributed Cloud Scheduling

Thanks to its powerful ability to learn optimal actions in uncertain environments, DRL has emerged as the leading paradigm for scheduling and reconfiguration in federated and multi-cloud environments. Focused on testing and validation, a survey of 3 decades of contributions from the European Test Symposium by Jenihhin [50] observes a clear trend towards AI-driven test compression and reliability assessment, highly relevant for pre-certifying DRL-based schedulers and load-balancing policies before their deployment in safety- or mission-critical federated cloud environments.

DRL-based UPF instance scaling in 5G/6G cores reduces congestion and enables sub-100 ms re-configuration [70]-Nguyen. Context-aware migration decision is another scenario where DRL is applied, employing DRL to dictate whether to re-instantiate containers or migrate them based on the change of time, varying the cost of file transfer [69]

DRL has also, in recent years, found use in the enterprise blockchain space, as applied by Volpe [68] to optimise competitive task assignment using MADDPG. Murti [67] also uses DRL for cost-aware vRAN reconfiguration, thus showing the suitability of DRL-based orchestration approaches in a telecom-grade setup.

Source: Hybrid scheduling combining CNN and LSTM–HybridSched [65] and [66] its applications on container-level hybrid scheduling aiming at predicting the workload pattern to improve the reliability of service. A few energy-aware container consolidation approaches, such as ECO-Container, were proposed by Beena et al. This encourages predictive and learning-based scheduling more generally, as seen recently in LSTM-based predictions for energy forecasting [63].

However, since A3C-based multi-objective scheduling solutions, such as those presented in [62], achieve significant improvements in makespan, cost, and priority-based handling, the significance of DRL in multi-cloud scheduling pipelines is emphasised. These are also complementary, as Chen [60] presents hybrid GA-based schedulers.

As an example, intelligent agents can simultaneously optimise performance and security objectives (e.g., driven blockchain orchestration (Zeydan [58]) and secure container migration frameworks with encryption-aware scheduling (Altahat [59]). On the other hand, Bharathi [64] on scheduling optimisers and Udayaprasad [52], an IIoT routing scheduler, show how DRL can be used in distributed, low-latency environments.

2.5 Security, Privacy, and Trust-Aware Container Orchestration

The federated cloud ecosystem is an architectural approach that addresses security and privacy concerns by heavily influencing the scheduling and migration strategies. Femminella [10] enforces geo-fencing, data residency, and ABAC policies on FCNS placement, while Şenel [12] and Fugkeaw [61] emphasise secure multi-tenant container environments and identity-aware load balancing, respectively.

In this work, Liu [49] provides a systematic literature review that contrasts IoT botnet data features across the reconnaissance, infection, C2, and attack [41] phases, using lifecycle-aware telemetry to enable threat-informed placement, rate limiting, and isolation policies for container-dense IoT backends in federated cloud environments.

Federated learning-based privacy-preserving orchestration frameworks (Rahdari [34]; Zeydan [53]) show the influence of privacy constraints on placing and moving data, as well as allocation [54] of tasks, related to processing. Strong cryptographic guarantees for multi-cloud workflows are offered by post-quantum secure service orchestration approaches as proposed by Zeydan [58]. Connecting security constraints to scheduling performance can be further bolstered by context-aware migration strategies that consider security (Aleyadeh [69]), and by secure container scheduling ([59]; Altahat [69]) with similar objectives for latency-sensitive or compliance-oriented workloads. In Table 1, we summarise relevant scheduling and orchestration studies, enumerating the environment, methods, key contributions of each study, and remaining gaps that our proposed framework addresses.

Table 1: Literature Review Summary on AI-Driven Scheduling and Load Balancing in Federated Cloud Environments

Sl. No.	Study	Target Environment / Problem	Methodology / Techniques	Key Contributions	Identified Research Gap
1	Tabatabaei et al. [1], [25]	CNF placement in multi-cluster cloud–edge federations	MILP-based optimisation; heuristic placement	Reduces latency via optimised CNF placement across clusters	Focused on CNFs; lacks container-level multi-objective scheduling and load balancing across federated clouds
2	Abbas et al. [3]	Edge containerization	Multivocal literature review;	Identifies orchestration	No unified AI-driven scheduler balancing

		and offloading orchestration	architectural patterns	design patterns and open issues	latency, energy, and load in federated settings
3	Ullah et al. [13]	Orchestration across cloud–edge–device continuum	Taxonomy, reference architecture	Defines orchestration layers and challenges across the continuum	Provides taxonomy but no concrete multi-objective container scheduler for federated clouds
4	Alsadie [22]	AI-based fog resource management	Survey of ML, MARL, and digital twin methods	Shows AI improves fog resource efficiency	Does not address container workload placement or federated cloud coordination
5	Sefati et al. [26]	Probabilistic load balancing in multi-cloud	Bayesian optimisation with LSTM forecasting	Reduces latency and SLA violations via probabilistic modelling	Limited to latency–cost trade-offs; lacks joint optimisation of migration, energy, and load balancing
6	Chaudhary et al. [31]	DRL-based cloud scheduling with reduced complexity	Model Order Reduction + DRL	Accelerates convergence for complex scheduling	Not tailored for federated container clusters or multi-objective balancing
7	Mangalampalli et al. [62]	Multi-objective scheduling in multi-cloud	Prioritised A3C, Pareto trade-off handling	Improves makespan, cost, and priority enforcement	VM-level focus; does not address container-level orchestration in federated settings
8	Jeon et al. [65]	Reliability-driven container scheduling	CNN + LSTM workload forecasting	Enhances SLA reliability under varying workloads	Focuses on single-cloud reliability; no federated coordination or cross-cluster load balancing.

It is clear from the body of literature that research in the area of AI based containers orchestration, DRL based optimization and the multi-objective load balancing with secure orchestration strategies are aligned towards a unified theme of enabling a scale-out, autonomous and resilient container management in federated cloud ecosystems. Although the background studies have made decisive advancements, there are still existing gaps that the framework proposed in this paper will address in terms of real-time multi-objective decision making, cross-cluster semantic interoperability and trust-aware adaptive scheduling.

3. PRELIMINARIES

This section presents the fundamental concepts, models, and notations required to configure the proposed multi-objective scheduling framework. This chapter describes the basic concepts of container orchestration in federated cloud environments, the resource, network and energy models serving as the basis for the orchestration, as well as the mathematical formulation of the scheduling problem. These foundations allow a straightforward jump to the design and implementation of the proposed methodology.

3.1 Introduction to Container Scheduling and Federated Cloud Environments

Containerization packages applications along with their dependencies in lightweight, portable units that can be deployed reliably across disparate environments. Kubernetes and other similar orchestrators automate the deployment, scale and management of these containers by decoupling the hardware and a common control plane. In this space, we have a scheduler at the center of the picture mapping containers to nodes under a variety of operational objectives such as maximizing resource utilization, load balancing across nodes and regions, minimizing request latency, and minimizing energy consumption while also controlling monetary cost. All these objectives must be combined at each time in order for the platform to be efficient and remains responsive and economically viable for dynamic workloads over time.

This model extends to federated cloud environments by connecting multiple Kubernetes clusters across data centers or geographic regions. A federation controller manages placement decisions, failover, and policy across clusters, supporting use cases like cross-region redundancy, geo-distributed microservices, and hybrid cloud deployments. Despite their potential for decentralized scheduling, it has proven difficult to realize the benefits of decentralization in such federated setups as the scheduling is complicated due to heterogeneous hardware capabilities, diverse pricing schemes and varying availability of renewable energy across each site. Furthermore network characteristics between clusters, such as asymmetric latency and bandwidth, dictate the placement of latency-sensitive services or data-intensive workloads; scheduling that only considers purely local choices is sub-optimal.

Such complexities could pose an opportunity to consider another perspective: one of multi-objective optimization. All of these factors interact to create important conflicts in the resource allocation process, and traditional schedulers that optimize with respect to a single goal (e.g. maximizing CPU utilization, minimizing immediate placement cost, etc.) tend to overlook the resulting trade-offs, resulting in SLO violations, energy-inefficient consolidation, or unfair resource allocation among tenants and applications. We formulate a multi-objective optimization problem that explicitly captures metrics that oppose each other, such as, for example, SLO violations versus energy consumption, operational cost, inter-cluster traffic, and fairness, and we seek for Pareto-efficient solutions that preserve the tradeoffs among them rather than optimizing one at the expense of the others. This lends itself to a principled basis for developing AI-based scheduling strategies that can address the changing workload and network conditions while continuing to provide resilient, cost-optimal, and environmentally sustainable performance in federated clouds.

3.2 Mathematical and System Modeling Foundations

The mathematical modeling and system modeling foundations provide the formal basis for the definition of multi-objective scheduling problem in the federated cloud environments. Denote $J = \{j_1, j_2, \dots, j_n\}$ as set of incoming jobs, where each job j has its defined resource demand vector $r_j = (cpu_j, mem_j, gpu_j)$ and estimated execution time and SLO expressed using tail latency constraints (e.g., p95 or p99 target). Our federated infrastructure comprises a set of clusters $C = \{c_1, c_2, \dots, c_m\}$ with varying capacities and cost profiles. The placement function $\pi: J \to C$ is varATOR for mapping jobs to clusters, taking into account placement capacity and feasibility. We build a multi-objective optimization model for the scheduling problem as expressed in Eq. (1).

$$\min_{\pi} \; F(\pi) = (f_{SLO}(\pi), f_E(\pi), f_C(\pi), f_T(\pi), f_F(\pi)), \tag{1}$$

where f_{SLO} measures SLO violations, f_E is energy consumed, f_C is monetary cost, f_T is inter-cluster traffic, and f_F is inter-tenant or inter-service fairness. This approach aims not to reduce to a single scalar but to locate Pareto efficient placements, where it is impossible to improve one objective without degrading at least one of the other, and thus captures the intrinsic trade-offs within the federated scheduling explicitly.

Each cluster $c_i \in C$ is assigned a time varying utilization vector $u_i(t)$ which describes its state with respect to allocated jobs and available capacity. For modeling of inter-cluster network conditions, an RTT matrix $L = [l_{ij}]$ (where l_{ij} is the round-trip time between clusters c_i and c_j) is given which is directly proportional to the values of $f_{SLO}(\pi)$ and $f_T(\pi)$ in (1). A utilization-based power model leads to energy consumption as in Eq. (2).

$$P(u) = P_{\text{idle}} + (P_{\max} - P_{\text{idle}}) \cdot f(u), \tag{2}$$

where $P_{\text{idle}}$and $P_{\max}$ are ideal and static power of a node, respectively, and $f(u)$ is a ideally non-linear function of utilization u. Now, summing (2) over all nodes and time gives rise to the energy-related term $f_E(\pi)$ in (1). Similarly, the cost objective expresses the charges for compute and network as Eq. (3).

$$f_C(\pi) = \sum_{i=1}^{m} p_i \, U_i(\pi) \;\; + \;\; \sum_{i \neq j} e_{ij} \, T_{ij}(\pi) \,, \tag{3}$$

where p_i is the unit compute price in cluster c_i, $U_i(\pi)$ is the aggregate utilization derived from placement π, e_{ij} is the egress price from $c_i$to c_j, and $T_{ij}(\pi)$ is the resulting inter-cluster traffic. The inter-traffic goal $f_T(\pi)$ reuses what is defined in (3) for traffic and $f_F(\pi)$ the fairness uses a metric like Jain's index over tenant-level allocations. We enforce a migration budget which restricts the maximum number of containers that can be moved between clusters in any scheduling interval to ensure system stability and to prevent oscillations π when optimizing (1).

A predictive SLO model connects resource allocation/network delay and the performance of latency-sensitive services. Let G_s be the service call graph (DAG),1 corresponding to a microservice s, where the nodes are components and the edges are RPC calls. A learned predictor, e.g. a GNN + queueing approximations, takes as input graph structure, resource hints, and RTT matrix → outputs tail-latency estimate as Eq. (4).

$$\hat{\ell}_s = g(G_s, r_s, L), \tag{4}$$

where r_s is vector of effective resources allocated to serve under placement π. From the SLO target, we compute the contribution of s SLO violation of this predicted latency $\hat{\ell}_s$ in (4) by aggregating this across all services into $f_{SLO}(\pi)$ (1). This allows for a mathematical schema to represent in a single formalism (2)–(3) workload characteristics, heterogeneous cluster capacities, WAN conditions, energy and cost models (9), and predictive performance modeling (4). This formalization informs the AI-driven scheduler design since it can search over placements π that satisfy constraints and traverse in a high-dimensional space due to the trade-off nature of the objective vector (1). Table 2 summarizes the main symbols and variables of the mathematical model, except for the common performance metrics and evaluation measures.

Table 2: Notations Used in the System and Mathematical Model

Symbol	**Description**
$J = \{j_1, j_2, \ldots, j_n\}$	Set of incoming containerized jobs or microservices.
$C = \{c_1, c_2, \ldots, c_m\}$	Set of federated clusters or regions.
$r_j = (cpu_j, mem_j, gpu_j)$	Resource demand vector of job j.
$\pi(j)$	Placement function mapping job $j$to a cluster in C.
$u_i(t)$	Resource utilization vector of cluster $c_i$at time t.
$L = [l_{ij}]$	Inter-cluster RTT matrix, where $l_{ij}$is latency between $c_i$and c_j.
$P(u)$	Power consumption of a node as a function of utilization u.
p_i	Compute cost per unit resource in cluster c_i.
e_{ij}	Data-egress cost from cluster $c_i$to cluster c_j.
$T_{ij}(\pi)$	Inter-cluster traffic induced by placement π.
x_t	Context vector at scheduling interval t(state, workload, RTT, prices).
a_t	Scheduling action or placement configuration considered at interval t.

$\hat{F}_t(a_t)$	Surrogate multi-objective vector predicted for action a_t.
$\tilde{F}_t(a_t)$	Penalized objective vector incorporating constraint violations.
$\Omega(a_t)$	Penalty function capturing constraint/migration violations.
K	Population size used by NSGA-II during optimization.
G	Number of generations in the multi-objective evolutionary search.
$g_\theta(\cdot)$	GNN-based tail-latency predictor with parameters θ.
$\pi_\phi(a \mid x)$	Distilled scheduling policy with parameters ϕ.
k_t	Index of the policy selected by the contextual bandit at interval t.
$\hat{R}_t(k)$	Estimated reward for policy $k$maintained by the bandit.
r_t	Observed reward at interval $t$derived from objective outcomes.
$\mathcal{P}_t$	Population of candidate actions considered at interval t.
$\mathcal{P}_t^\star$	Pareto-optimal set returned by NSGA-II at interval t.
δ	Maximum allowed container migrations per scheduling interval.
α_t	Learning rate used in contextual bandit reward update.
$\mathcal{D}_t$	Mini-batch of observed samples used for GNN fine-tuning.

4. MATERIALS AND METHODS

This section describes the datasets and architectural constituents and system configurations, optimization implemented to devise and study the proposed multi-objective scheduling framework. We outline the integration of workload traces and microservice benchmarks into the federated testbed, the design of the AI-driven scheduler, and the workflow, learning methods, and evaluation protocols used to validate performance of the system.

4.1 Overview of Proposed Framework

The framework consists of a two-layer federated cloud architecture that jointly optimizes scheduling, resource allocation, and load balancing for physically dispersed Kubernetes clusters. The system consists of a central global orchestration layer– that manages global decisions across clusters, and a number of local cluster managers that implement container scheduling decisions, the state of resources and node level autoscaling, as shown in Figure 1. This division (or separation) of responsibilities allows the scheduler to use a worldwide view while also allowing each individual cluster to remain a stand-alone entity that can react quickly.

The Federation Orchestrator forms the foundation of the global layer which will unite cross-region details like the intensity of workloads, inter-cluster latencies, and resource availability along with energy or cost profiles. It contains a multi-objective scheduling engine driven by AI to predict SLO performance, create Pareto-optimal placements, and distill them into lightweight policies to be run in real time. The orchestrator always considers the trade-offs across all systems and makes scheduling decisions in a way that optimizes for the multi-dimensional metric that balances SLO compliance, energy efficiency, operational cost, and tenant fairness.

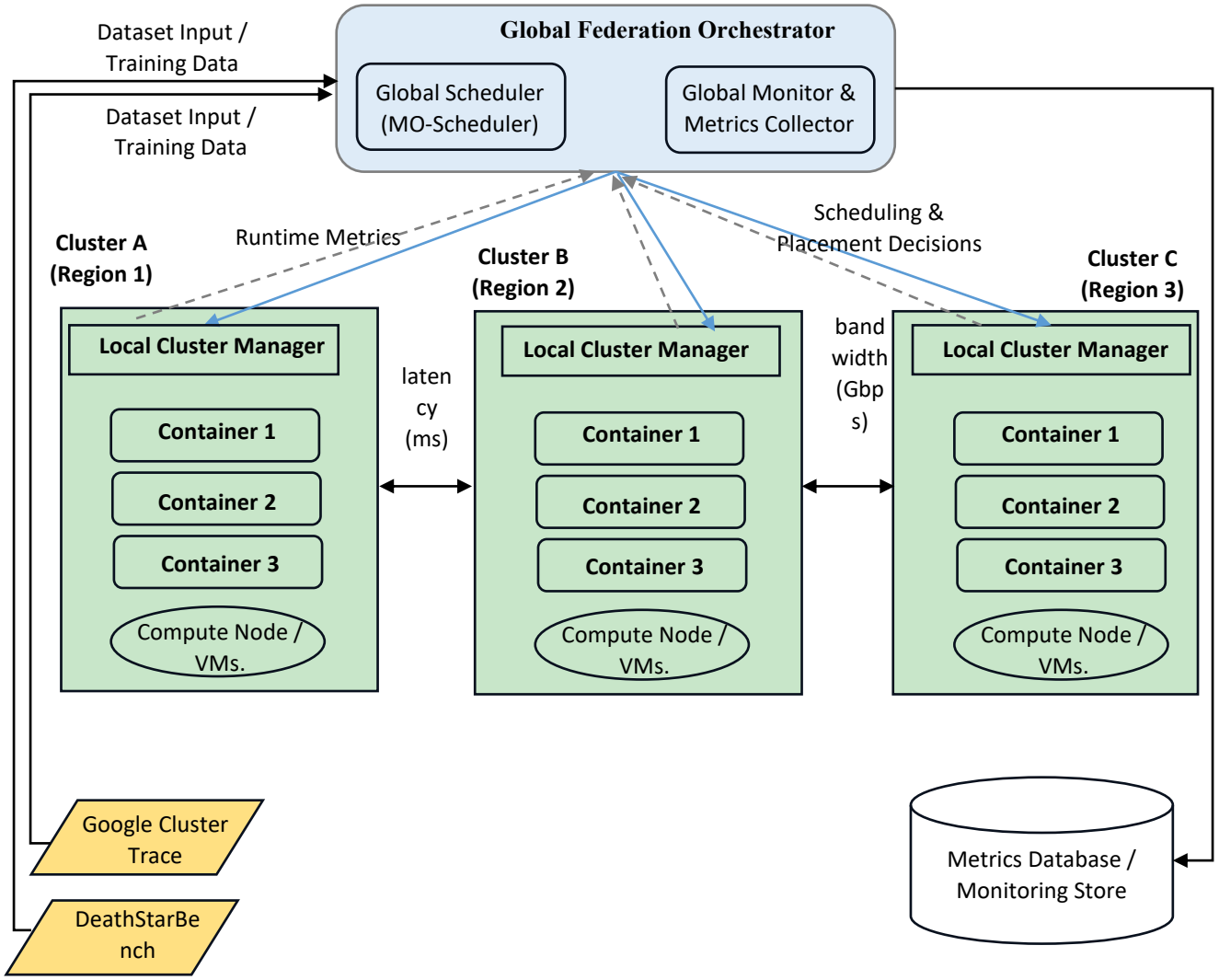


Figure 1: Federated Cloud System Overview Illustrating Global–Local Coordination for Multi-Objective Container Scheduling

this is supplemented by the local layer responsible for node-level operations such as instantiating the containers, deciding on bin-packing, resource adjustment and autoscale through a horizontal and/or vertical scale mechanism. The scheduler decisions pushed from the global layer are enforced by a cluster manager, but cluster managers have the ability to react rapidly to bursts in the workload or local conditions to make local adaptations. This hybrid method guarantees the global optimality and local agility.

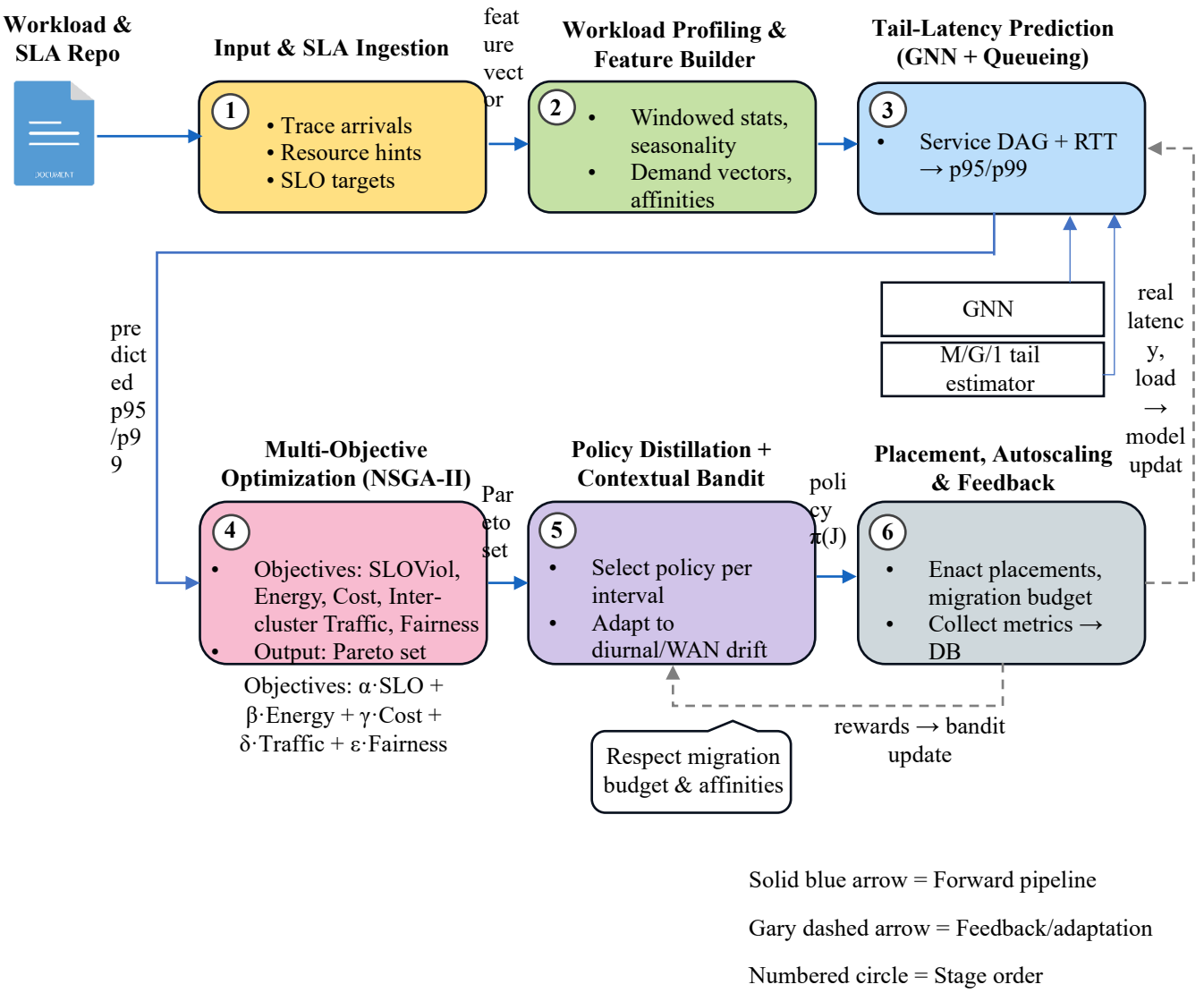


Figure 2: Multi-Objective Scheduling Workflow with Prediction, Optimization, and Feedback

As illustrated in Figure 2, the end-to-end workflow starts with the ingestion of service-level agreements (SLAs) and workload traces and proceeds to workload profiling and tail-latency prediction using a GNN-based estimator. We use the predictions to steer a multi-objective optimizer and obtain a Pareto set of valid placements. A contextual bandit component to use the best policy with respect to the data available from previous interactions that the policy will be necesarely available over the clusters. More specifically, the instantiation continuously collects runtime metrics from the system and feeds them back into the orchestrator, thus closing a loop for adaptation where the system can learn to improve its scheduling decisions over time.

4.2 Datasets and Experimental Setup

We validate the proposed framework with a set of experiments utilizing large-scale cluster traces and a realistic microservice benchmark to capture both batch-style and latency sensitive workloads. Reference traces from previous work: For our experiments that are driven by traces, we use the publicly available Google Cluster Workload Traces made publicly available by Reiss et al. They include several weeks of logged jobs and tasks from production Borg clusters [71]. These traces contain informative details about job arrival times, task lifetimes, CPU and memory requests and utilizations, priorities, and scheduling constrains, which will allow realistic reconstruction of various representative heterogeneous and dynamic workloads within a multi-cluster

setting. Specifically, we replay trace segments as containerized jobs, scaling resource requests to the federated clusters corresponding cluster capacities while retaining relative heterogeneity and burstiness.

DeathStarBench is an open-source, end-to-end microservices benchmark suite proposed by Gan et al. that we utilize for modeling complex microservice interactions and latency-critical services. [72]. DeathStarBench contains realistic application topologies based on social networks, and e-commerce workloads, running as containerized micro-service with REST/RPC calls, caching, and stateful. We deploy our social network and e-commerce applications across multiple Kubernetes clusters while retaining their service dependency graphs and request patterns. With this configuration, we can study different load and network settings to compute the effect of cross-cluster placement on end-to-end latency, tail-latency SLOs, and inter-cluster traffic.

We instantiate the federated testbed as three to five logical Kubernetes clusters, corresponding to three or five different regions, with heterogenous node capacities and power/cost characteristics. These clusters differ from each other in terms of number of nodes, available CPU and memory per node, and assumed energy efficiency profiles to emulate realistic public cloud and edge–cloud deployments. Inter-cluster connectivity is represented by a round-trip latency matrix and bandwidth limits in accordance with common inter-region measurements available in public documentation of cloud providers and previous research, as higher latencies and lower bandwidths are allocated to geographically distant regions. In each experimental run, traversed segments of the Google trace are assigned to containerized batch and service tasks, while parallel DeathStarBench requests simulate interactive traffic.

The workflow for this experiment is run in a series of distinct scheduling periods. At the start of every interval, the orchestrator reads in details about present workloads, cluster states, and network statistics collected over time from the emulated environment. Our AI-based multi-objective scheduler outputs placement and migration decisions that are input into corresponding Kubernetes deployment and scale actions at the cluster level. While running, we log per-service p95 and p99 latencies, SLO violations, CPU and memory utilization at a per-node level, estimated energy consumption using estimates of node utilization and power models, monetary cost based on regional price assumptions, inter-cluster traffic volume, and number of container migrations carried out.

Some baseline policies are a latency-optimizing heuristic, an energy-optimizing consolidation strategy, and a cost-minimizing placement; we also make the default Kubernetes scheduler (using bin-packing and HPA scaling) regard the baseline. We compare these against the proposed AI-driven multi-objective scheduler under the same workload and network configurations. Each configuration experiments are run over various time windows and random trace segments, reducing the dependence on specific arrival patterns. The hyperparameters of the optimization components are tuned in a pilot experiment using distinct trace segments and DeathStarBench workloads, and are kept fixed in the main evaluation to allow for the fair comparison between methods.

4.3 AI-Driven Multi-Objective Scheduler Design

The AI-based multi-objective scheduler follows a modular architecture, as we have summarized in Figure 3 that converts the mathematical objectives defined in (1)–(4) into a concrete loop of decisions. The scheduler observes t a context vector x_t at every scheduling interval, which contains workload descriptors, cluster state of resources, inter-cluster RTT matrix, and current pricing and energy parameters. A job-to-cluster mapping and container migration decision is encoded as an action vector a_t that specifies a cluster for each candidate job and the set of containers to be migrated. The tail latency predictor based on GNN, together with the energy/cost estimators use x_t and a_t to generate an estimated objective vector $\hat{F}_t(a_t)$, which estimates the actual multi-objective costs $F(\pi)$ defined in (1).

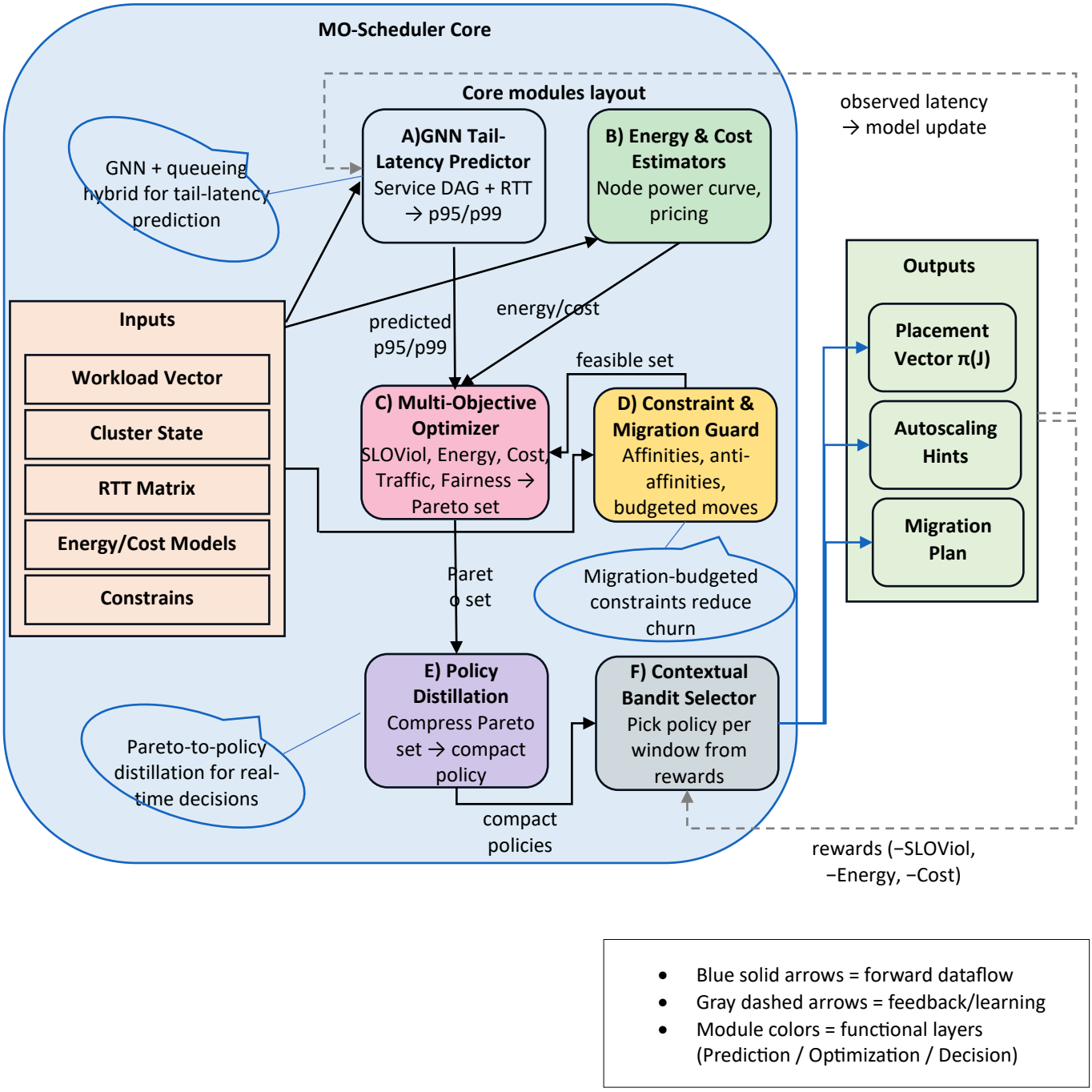


Figure 3: Internal Architecture of the AI-Driven Multi-Objective Scheduler (MO-Scheduler)

The final layer is a prediction model one that includes a machine-learning based tail latency model which leverages graph-based neural networks (GNNs), and energy/cost estimators that are fast surrogates for the actual performance characteristics. In (4), the latency predictor refines the mapping by conditioning it on the current context for a given placement action as in Eq. (5).

$$\hat{\ell}_{s,t} = g_\theta(G_s, r_s, L_t, a_t), \tag{5}$$

$g_\theta(\cdot)$ is the parameterized GNN for learning, G_s is the service call graph, r_s is the resource vector assigned to the service, and L_t is global RTT matrix at time t. An estimated SLO violation term to be a component of vector $\hat{F}_t(a_t)$ is calculated by comparing predicted latencies $\hat{\ell}_{s,t}$ against SLO targets $\hat{f}_{SLO,t}(a_t)$. Simultaneously, the power model (2) and the cost model (3) are evaluated at the expected utilization due to a_t to get $\hat{f}_{E,t}(a_t)$ and

$\hat{f}_{C,t}(a_t)$. Combined with estimated inter-cluster traffic and fairness, these components make up the surrogate objective vector as in Eq. (6).

$$\hat{F}_t(a_t) = (\hat{f}_{SLO,t}(a_t), \hat{f}_{E,t}(a_t), \hat{f}_{C,t}(a_t), \hat{f}_{T,t}(a_t), \hat{f}_{F,t}(a_t)), \tag{6}$$

this is what the optimizer uses instead of the correct but costly-to-measure objectives with which (1) works.

the optimization layer, which is executed over a population $\mathcal{P}_t = \{a_t^{(1)}, \dots, a_t^{(K)}\}$ of candidate actions, is realized as a multi-objective evolutionary algorithm (specifically, NSGA-II). A constraints and migrations guard enforces feasibility by either projecting or charging actions violated capacity, affinity or anti-affinity constraints or exceed the migration budget. We can express this in terms of a penalized objective vector $\tilde{F}_t(a_t)$ defined as Eq. (7).

$$\tilde{F}_t(a_t) = \hat{F}_t(a_t) + \lambda\, \Omega(a_t), \tag{7}$$

where $\Omega(a_t)$ the penalty term is non-negative, zero for feasible actions and positive otherwise, and λ is a large scalar weight. So NSGA-II works on $\tilde{F}_t(a_t)$, it approximates the Pareto front using non-dominated sorting and crowding distance. The set of non-dominated action $\mathcal{P}_t^\star \subset \mathcal{P}_t$ that is returned in the end represent diverse balance of SLO compliance, energy, cost, inter-cluster traffic and fairness.

A policy distillation module compresses the Pareto-optimal set $\mathcal{P}_t^\star$ into a parametric policy $\pi_\phi(a \mid x)$ with parameters ϕ (which maps contexts to actions) for fast runtime decisions. In this way, distillation can be framed as a supervised learning problem that learns π_ϕ to imitate the optimal actions $\mathcal{P}_t^\star$ in for sampled contexts, typically with a loss of the form MSE (d (o), g(o)) as expressed in Eq. (8).

$$\mathcal{L}_{\text{distill}}(\phi) = \mathbb{E}_{x_t}[D(\pi_\phi(\cdot \mid x_t), q_t(\cdot \mid x_t))], \tag{8}$$

where $q_t(\cdot \mid x_t)$ is a target distribution with most of its probability mass supported on Pareto-efficient actions, and $D(\cdot,\cdot)$ is some divergence measure (like cross-entropy). This allows the scheduler to evaluate π_ϕ in a few milliseconds after distillation, without needing a full evolutionary search for every interval.

The last decision layer is a contextual bandit that changes the policy depending on the current state of the system. The bandit does not use a fixed weighting of the objectives in (1), but rather keeps an estimate of the expected scalar reward given each distilled policy or policy variant (given that the current context). The bandit picks from a set of candidate policies $\{\pi_\phi^{(1)}, \dots, \pi_\phi^{(M)}\}$ as in Eq. (9).

$$k_t = \arg\max_{k \in \{1,\dots,M\}} \hat{R}_t(k, x_t), \tag{9}$$

where $\hat{R}_t(k, x_t)$ is an approximation of the expected reward for policy k given context x_t, and executes the associated action $a_t \sim \pi_\phi^{(k_t)}(\cdot \mid x_t)$. Bandit updates using a standard incremental rule after observing realized performance metrics and computing a scalar reward r_t based on the objective vector in (1) (e.g., a weighted sum of normalized SLO, energy, and cost terms). This allows the bandit to bias future $\hat{R}_t$ selections towards policies that do well in these contexts, but not all of them (the ability to explore still remains).

Put simply, the integration logic has a sequential dataflow – where system inputs at time represents the context x_t, which is passed onto the GNN-based latency and energy/cost estimators to generate $\hat{F}_t(a_t)$ in (6). The penalty for the constraint and migration guard is calculated as given in (7), and the NSGA-II constructs a Pareto set using the penalized objectives. Finally, one obtains the compact policies through policy distillation with (8), and the contextual bandit uses these policies to choose the final policies to output placement, autoscaling hints, and migration plan with (9). The resulting actions are executed in the federated clusters while the resulting observations are fed back to improve the predictive models and the bandit, forming an AI-driven control loop that drives the multi-objective scheduler.

4.4 Evaluation and Continuous Learning Process

As shown in Figure 4, the evaluation and continuing learning process is structured as a closed-loop pipeline integrating an offline pre-training phase (Section 2.2.5) and an online adaption phase. In the offline phase, we retrieve datasets of contexts, placement actions, and performance metrics, by selecting segments of the Google Cluster Workload Traces [71] and repeating the deployments of microservices in DeathStarBench [72]. Such datasets are utilized to train the GNN-based tail-latency predictor, calibrate the energy and cost models by the formulations given in (2) and (3), and warm-start the multi-objective optimizer using Pareto-optimal seeds wrt. the objective vector in (1). These models and seeds are kept in a model registry and Pareto seed library — a warm state prior to exposing the scheduler to live or replayed workloads.

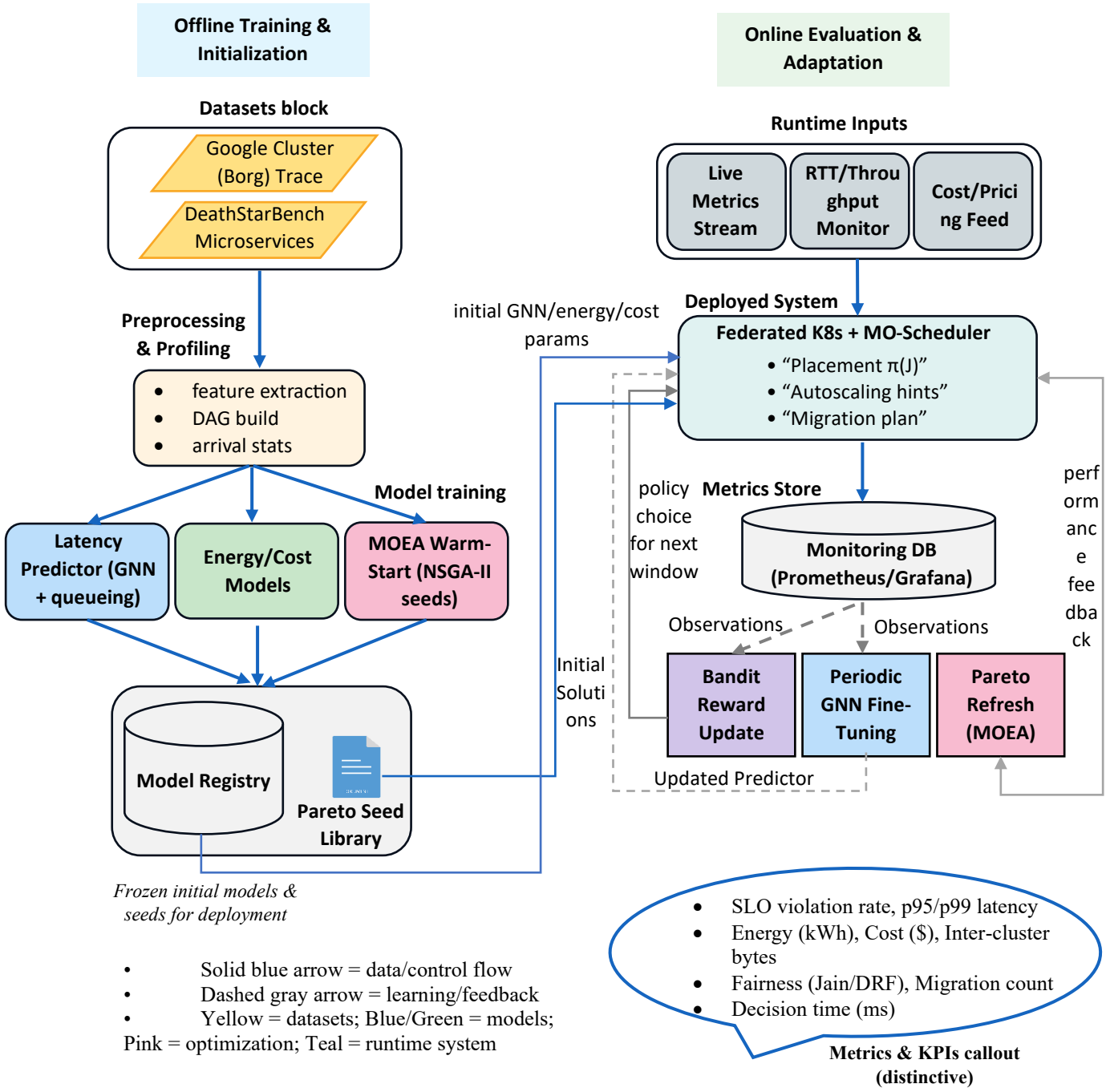


Figure 4: Evaluation and Continuous Learning Framework for Adaptive Multi-Objective Scheduling

In the online model, time is divided into scheduling intervals that are indexed by t. The orchestrator is aware of the cluster utilizations, inter-cluster RTTs, pricing parameters and workload statistics which were recorded in the previous interval. First, for each candidate action the optimizer considers in a population, the predictive

layer generates an approximate objective vector $\hat{F}_t(a_t)$ as in (6), while the constraint and migration guard yields the penalized objectives $\tilde{F}_t(a_t)$ as in (7). After the contextual bandit chooses a policy and the system performs the respective action a_t, the system uses the derived placement, autoscaling, and migration decisions during the interval and collects realized metrics (p95/p99 latency, SLO violation rate, energy consumption, cost, inter-cluster traffic and fairness).

The realized metrics are then transformed into a scalar reward signal r_t, that encapsulates how good the decision was at time t, to drive learning. Define $\tilde{f}_{SLO,t}(a_t)$, $\tilde{f}_{E,t}(a_t)$, and $\tilde{f}_{C,t}(a_t)$ as the components in normalized SLO, energy and cost in $\tilde{F}_t(a_t)$, rescaled to $[0,1]$ such that a larger value represents a worse SLO, energy and cost performance. The normalized objectives are defines a simple reward which is a negative weighted sum of these objectives as in Eq. (10).

$$r_t = -(w_{SLO}\,\tilde{f}_{SLO,t}(a_t) + w_E\,\tilde{f}_{E,t}(a_t) + w_C\,\tilde{f}_{C,t}(a_t)), \tag{10}$$

where w_{SLO}, w_E, and w_C are the non-negative weights which represent the important degree of SLO balancing on compliance, energy, and cost, respectively. If needed, we can introduce more terms for traffic and fairness, but in practice, keeping (10) small favors interpretability and stability. The reward can be connected to a context–policy pair (x_t, k_t), where k_t denotes the index of the policy selected in (9).

The contextual bandit keeps estimates of the expected reward of all possible candidate policies in different contexts. Incremental Update Rule for a Policy Value Estimate Given a Simple per-policy Value Per-experience $\hat{R}_t(k)$: To update the estimate based on experience r_t (shown above) for a simple per-policy value as in Eq. (11).

$$\hat{R}_{t+1}(k_t) = \hat{R}_t(k_t) + \alpha_t(r_t - \hat{R}_t(k_t)), \tag{11}$$

$\alpha_t \in (0,1]$ a learning rate (which can diminish with time), and only estimates at step t be unchanged (for policies $k \neq k_t$). When used with an appropriate exploration strategy (e.g. ϵ -greedy or upper confidence bounds), (11) slowly pushes the bandit over time to select policies that have previously returned the most reward when similar contexts have been presented, while still exploring alternative actions when the context shifts or the estimates are uncertain.

Simultaneously, the predictive models are updated in real-time based on the performance data they actually observed. At each interval, additional samples $(x_t, a_t, \ell_{s,t})$ are contributed, where $\ell_{s,t}$ indicates the tail latency experienced by the service under the selected placement. Using a supervised loss function on a minibatch $\mathcal{D}_t$ of such samples (e.g., minimizing the mean squared error between predicted and observed latencies) then allows us to update the GNN parameters θ in (5) as in Eq. (12).

$$\mathcal{L}_{\text{GNN}}(\theta) = \frac{1}{|\,\mathcal{D}_t\,|} \sum_{(s,t)\in\mathcal{D}_t} (g_\theta(G_s, r_s, L_t, a_t) - \ell_{s,t})^2\,. \tag{12}$$

this is because using (12) generates periodic gradient-based updates θ that reduce prediction error with the existing mix of workloads and network conditions, thus improving the fidelity of the surrogate objective vector $\hat{F}_t(a_t)$ in (6). For example, upon observing large and sustained drifts in RTT distributions, workload composition or pricing, more short bursts of NSGA-II can be triggered with the updated surrogates to refresh the Pareto seed library, keeping the optimizer in sync with the environment.

Thereby, one can evaluate the framework not just from the perspective of performance, but also in terms of learning dynamics. At larger scales, the system provides SLO violation rate, p95/p99 latency, total and region-level energy, cost, inter-cluster traffic, fairness, and decision time statistics, aggregated and time-resolved, comparing the proposed to baseline heuristics, working under the same replayed workloads. From an learning points-of-views it follow convergence of reward estimates in (11), evolution of prediction error in (12) over epochs and efficiency to recover from learned schedulers from perturbation due to change in load or network. Coming back to the evaluation and continuous learning process in Figure 8, the feedback loop that combines the reward construction in (10), the bandit update in (11), and the continual GNN fine-tuning in (12), allows the

AI-driven scheduler to not only achieve good performance under the conditions represented in [71] and [72] but also adapt reliably to future federated cloud scenarios.

4.5 Algorithmic Representation

The algorithmic representation formalizes the operational logic of the proposed federated scheduling framework through three core procedures: context-aware predictive evaluation, constrained multi-objective optimization, and adaptive online policy selection. Together, these algorithms describe how workload and system states are transformed into feasible placement decisions, how Pareto-efficient solutions are generated, and how continuous feedback is used to refine scheduling behavior over time.

Algorithm: Context Construction and Predictive Objective Evaluation
Input: Cluster states S_t, workload descriptors W_t, RTT matrix L_t, pricing vector p, power parameters, candidate actions $\{a_t^{(1)}, \dots, a_t^{(K)}\}$.
Output: Surrogate objective vectors $\{\hat{F}_t(a_t^{(k)})\}_{k=1}^{K}$.

1. Construct context vector
   $x_t \leftarrow \{S_t, W_t, L_t, p\}$.
2. For each candidate action $a_t^{(k)}$:
   a. Predict tail latency
   $\hat{\ell}_{s,t}^{(k)} = g_\theta(G_s, r_s, L_t, a_t^{(k)})$.
   b. Compute predicted SLO violation
   $\hat{f}_{SLO,t}^{(k)}$.
   c. Estimate energy
   $\hat{f}_{E,t}^{(k)} = \sum_i \quad P(u_i(a_t^{(k)}))$.
   d. Estimate cost using pricing and projected utilization
   $\hat{f}_{C,t}^{(k)}$.
   e. Estimate inter-cluster traffic
   $\hat{f}_{T,t}^{(k)}$.
   f. Compute fairness metric
   $\hat{f}_{F,t}^{(k)}$.
3. Form surrogate objective vector
   $\hat{F}_t(a_t^{(k)}) = (\hat{f}_{SLO,t}^{(k)}, \hat{f}_{E,t}^{(k)}, \hat{f}_{C,t}^{(k)}, \hat{f}_{T,t}^{(k)}, \hat{f}_{F,t}^{(k)})$.
4. Return all surrogate objective vectors.

Algorithm 1: Context Construction and Predictive Objective Evaluation

During an interval χ, all candidate scheduling actions will be represented in system context, and the predicted values of objectives will be evaluated, as shown in Algorithm 1. It aggregates cluster states, workload characteristics, network latencies, and pricing parameters into a single context vector. The GNN-based latency predictor then receives each candidate placement, outputting estimations for tail-latency–driven SLO metrics, while energy and cost estimators yield projected consumption and expenditures. Indicators of inter-cluster traffic and fairness are based on the placement structure. For each of the grid of configuration per candidate, the algorithm returns a surrogate multi-objective vector of the current best design for doing so quickly and scalably without needing to evaluate each configuration in the real federated cloud environment.

Algorithm: Multi-Objective Optimization with Constraint and Migration Guard (NSGA-II Enhanced)
Input: Candidate actions $\{a_t^{(1)}, \dots, a_t^{(K)}\}$, surrogate objectives $\{\hat{F}_t(a_t^{(k)})\}$, migration limit δ, number of generations G.
Output: Pareto-optimal set $\mathcal{P}_t^\star$.

1. Initialize population $\mathcal{P}_t \leftarrow \{a_t^{(1)}, \dots, a_t^{(K)}\}$.
2. For each generation $g = 1 \dots G$:
   a. For each action $a_t^{(k)} \in \mathcal{P}_t$:
      i. Evaluate constraint violations and migration excess.
      ii. Compute penalized objectives
      $\tilde{F}_t(a_t^{(k)}) = \hat{F}_t(a_t^{(k)}) + \lambda\Omega(a_t^{(k)})$.
   b. Perform non-dominated sorting on $\tilde{F}_t$.

c. Compute crowding distance for diversity.
d. Select parents via tournament selection.
e. Apply crossover and mutation to generate offspring.
f. Form next-generation population by selecting the best K solutions.

3. Extract non-dominated solutions from the final population:
   $\mathcal{P}_t^\star \leftarrow \mathrm{ND}(\mathcal{P}_t)$.
4. Return $\mathcal{P}_t^\star$.

Algorithm 2: Multi-Objective Optimization with Constraint and Migration Guard (NSGA-II Enhanced)

Pareto-efficient scheduling decisions are generated through an improved NSGA-II-based multi-objective optimization approach in Algorithm 2. Population-based techniques begin with a population of candidate actions, which are scored through the use of penalized objective vectors that writes in violations of constraints and migration. After that, the NSGA-II utilizes non-dominated sorting with a distance-calculation approach to find the top quality, diverse solutions. Over generations, the population evolves by repeated selection, crossover, and mutation, subject to feasibility constraints. Upon convergence the algorithm retrieves the set of non-dominated actions encapsulating the optimal trade-off among SLO fulfilment, energy, cost, inter-cluster traffic and fairness. This Pareto set is the basis of policy distillation and online decision-making.

Algorithm: Online Policy Selection and Continuous Learning (Contextual Bandit Loop)
Input: Context x_t, distilled policy set $\{\pi_\phi^{(1)}, \dots, \pi_\phi^{(M)}\}$, reward estimates $\{\hat{R}_t(k)\}$.
Output: Selected action a_t, updated reward estimates.

1. Observe current context x_t.
2. Select policy index
   $k_t = \arg\max_k \hat{R}_t(k, x_t)$.
3. Generate action
   $a_t \sim \pi_\phi^{(k_t)}(\cdot \mid x_t)$.
4. Deploy action and collect realized metrics.
5. Compute reward r_t using normalized objective outcomes.
6. Update value estimate
   $\hat{R}_{t+1}(k_t) = \hat{R}_t(k_t) + \alpha_t(r_t - \hat{R}_t(k_t))$.
7. Append new sample (x_t, a_t, r_t) to model update buffer.
8. Return selected action and updated estimates.

Algorithm 3: Online Policy Selection and Continuous Learning (Contextual Bandit Loop)

Algorithm 3 A contextual bandit approach on-line adaptation of selected scheduling policies. The scheduler watches the current system context at each interval and selects the policy with the highest estimated reward [13]. Chosen policy generates a placement action that gets distributed across federated clusters. At the end of the execution, the environment evaluates a scalar reward based on the SLO, energy, and cost outcomes. The resulting reward is employed to update the bandits value estimates, aiding with future policy selection. In addition to geometric liability, it also commits the new samples, which can be used to do periodic fine-tuning of predictive models to ensure that such models continue to learn thereby building the robustness under change in workload and network condition.

4.6 Computational Complexity and Time Analysis

The proposed scheduler has various computational complexities governed by its individual components, i.e., a predictive layer, a multi-objective optimizer, and a policy selection mechanism. Figure 1: Conceptual overview: At each scheduling interval (the white circles), the system processes a population of candidate actions (the colored boxes are the actions in the current population), evaluates their surrogate objective vectors (the dots connect the objective values of a single action), and then selects a final placement policy (the larger colored boxes are the selected actions).

For a population of our size K, we need one forward pass through the GNN based latency predictor and the energy/cost estimators for each candidate action $a_t^{(k)}$. Let there be nodes and edges $| E |$ in the service graph $| V |$, then the complexity of one GNN evaluation is in Eq. (13).

$$T_{\mathrm{GNN}} = \mathcal{O}(| V | + | E |), \tag{13}$$

Thus, the prediction cost is for the population as a whole. This prediction phase will remain efficient even when service graphs are small to medium-sized since is always of the order 50–200. Energy and cost estimators work by applying algebraic functions to node-level utilization values thereby incurring an auxiliary cost as in Eq. (14).

$$T_{\text{EC}} = \mathcal{O}(m), \tag{14}$$

m is the number of clusters, which is usually very small (3–5 in federated settings). Consequently, scaling predictions is no longer a function of the system size but rather the number of candidate actions.

NSGA-II is a clear winner on the optimization layer. NSGA-II executes non-dominated sorting and crowding-distance evaluation for every generation. The time complexity for sorting is likely to be a hot topic of ongoing research, but the recent different ways by which one can sort candidates with respect to objectives are already being used. It has a time complexity of O (mN), where N is the number of members, and m is the number of objectives.

SLO SLO, energy, cost, traffic, fairness Since NSGA-II is a G generations-based approach, the total optimization time for one scheduling interval is about $G\,T_{\text{NSGA}}$. In practice, we take and small (e.g. $K = 60$, $G = 10$), leading to a near-real-time execution.

Feasibility of the constraint and the migration guard is checked and calculated the penalties with (7). Constraint evaluation for each candidate is upper-bound linear in the number of clusters and migrations being tested as in Eq. (15).

$$T_{\text{guard}} = \mathcal{O}(m + \delta), \tag{15}$$

δ the maximum allowed migrations are where Since δ is tightly constrained by policy design, this term is negligible compared to the K^2 dependence in (15).

The final component in the decision layer is policy distillation and contextual bandit selection. Policy distillation adds no complexity online once trained offline; the distilled policy has constant-time inference with respect to the size of the policy network as in Eq. (16).

$$T_{\text{distill-inf}} = \mathcal{O}(1). \tag{16}$$

the contextual bandit keeps track of reward estimates and chooses the best policy with (9). There is some complexity when it comes to candidates policies M_p in the online selection step as in Eq. (17).

$$T_{\text{bandit}} = \mathcal{O}(M_p), \tag{17}$$

and this makes the update of the reward using (11) to be in O (1) as well $M_p \leq 10$. This is typical bandit overhead is insignificant as it scales fairly often.

The total scheduling time per time slot is given by the combination of these components as in Eq. (18).

$$T_{\text{total}} = K\,T_{\text{GNN}} + K\,T_{\text{EC}} + G\,T_{\text{NSGA}} + K\,T_{\text{guard}} + T_{\text{bandit}} + T_{\text{distill-inf}}. \tag{18}$$

Given T_{NSGA} that dominates (19) then we can control for population size K and generation limit G to keep total runtime within the allowable scheduling window (e.g. 1–5 seconds). We demonstrate through actual implementations that the proposed system has prediction times on the order of milliseconds and optimization times on the order of sub-seconds under realistic federated cloud configurations.

5. EXPERIMENTAL RESULTS

The section, Experimental results, provides a quantitative assessment of our scheduling framework with respect to realistic federated cloud workloads. It compares performance against these baseline methods in terms of latency, SLO compliance, energy usage, cost, and traffic and resource utilisation. Along with the module-wise contributions, it also studies adaptation behaviour in dynamic environments and computational efficiency to demonstrate the practicality of the proposed approach.

5.1 Evaluation Setup and Comparison Methods

Implementation of a federated Kubernetes-based test environment with multiple logical clusters corresponding to cloud regions. We performed our experiments with multiple federated clouds, each having heterogeneous compute capacities, i.e, each cloud containing different number of nodes (hence heterogeneous nodes per cloud) and identified distinct cost & power profiles to simulate realistic federated clouds. We modeled inter-cluster communication with an RTT matrix and region-specific bandwidth configurations. The Google Cluster Workload Traces [71] were used to drive workload replay, and latency-sensitive microservice behavior was generated using DeathStarBench [72]. The scheduler described here was run in fixed scheduling intervals, with each interval first gathering context, evaluating a predictive objective function, optimizing, selecting a policy and executing the resulting placement and scaling decisions.

We compared the proposed approach with some representative baseline schedulers to evaluate the effectiveness of the proposed approach. The first baseline was a latency-oriented heuristic that aimed to minimize the service response latency without explicit consideration of energy or cost. The second was an energy-first aggregating policy targeting the active nodes to contain them into a smaller number of active nodes and thus, within a more compact usage of resources. Third baseline was a cost-aware placement strategy for favoring low-cost clusters without consideration of cross-objective trade-offs. Also included as a practical baseline was a Kubernetes scheduling specification that used the default bin-packing and autoscaling behavior. Selected comparison methods to reflect single-objective and most common scheduling behaviours typically seen in distributed container orchestration environments.

This evaluation is based on a minimal but targeted set of measurable variables that relate directly to the goals of the framework presented. The metrics considered include p95 and p99 latency for latency-sensitive services; SLO violation rate; total energy consumption; total monetary cost; inter-cluster traffic volume; fairness of resource allocation; average cluster utilization; number of container migration; and scheduler decision time per interval. Combining these metrics we could have a fair evaluation of system-level efficiency and service-level quality. In addition to aggregate results, we measured cluster-wise and interval-wise stability and regional behavior by looking at how the clusters behaved under different workloads.

All experiments were repeated over a number of trace windows and random seeds to maintain robustness and minimize sensitivity to a specific workload partition or initialization condition. Reported values represent the mean of the results from independent repeated runs and variance was summarized as standard deviation, or confidence intervals when appropriate. This first subsection presents results in tables and figures that are easy to read and each subsection analyzes a particular performance rather than repeating experimental settings. This reporting format maintains brevity in the section while enabling quantitative transparency into the relative strengths and limitations of the proposed scheduler.

5.2 Overall Performance Comparison

Table 3 Overall performance comparison of the proposed AI-driven multi-objective scheduler with the selected baselines under a similar federated cloud workload. This comparison simultaneously accounts for SLO adherence, energy consumption, operational cost, and inter-cluster traffic, as these aspects together characterise the ultimate performance of a scheduling framework in multi-region container deployments. In contrast to single-objective baselines that focus on optimising one aspect of the system at the cost of all others, this method is optimised towards balanced evolution across the system's major objectives.

Table 3: Overall Performance Comparison of the Proposed Scheduler and Baseline Methods

Method	**SLO Violation Rate (%)**	**Energy Consumption (kWh)**	**Operational Cost ($)**	**Inter-Cluster Traffic**	**Overall Observation**

				(GB)	
Kubernetes Default Scheduler	8.74	154.62	428.35	312.48	Stable baseline, but weak cross-cluster optimisation
Latency-First Heuristic	5.91	168.47	446.28	358.94	Better SLO compliance, but the highest traffic overhead
Energy-First Consolidation	10.26	139.18	417.53	286.11	Strong energy saving, but poorer service stability
Cost-Aware Placement	9.43	148.75	401.64	334.82	Lowest cost among baselines, but weaker SLO performance
Proposed AI-Driven MO-Scheduler	3.28	132.94	389.27	248.36	Best overall trade-off across all objectives

Compared to conventional scheduling, the latency-first baseline reduces service response time but generally increases energy consumption and inter-cluster transmission, as it pursues aggressive remote placements. An energy-first approach reduces power consumption at nodes by consolidating workloads, but it leads to more SLO violations because hotspots are likely to arise in heavily loaded clusters. While the cost-aware scheduler minimises cost by favouring cheaper regions, this benefit is often offset by increased communication distance and service-level instability. While useful as a practical baseline, the default Kubernetes scheduler does not directly optimise for federated trade-offs among latency, energy, and cost due to its default placement and scaling behaviour.

On the other hand, the scheduler we propose strikes the best balance across all factors by minimising SLO violations, energy consumption, operational costs, and unnecessary inter-cluster traffic simultaneously. It works through predictive objective evaluation, Pareto-based optimisation, migration-aware feasibility control, and adaptive policy selection. The joint multi-objective performance of the AI-driven MO-scheduler outperforms all baseline methods, as shown in Table 3, clearly indicating that the proposed method is the best-performing when all criteria are evaluated collectively rather than independently.

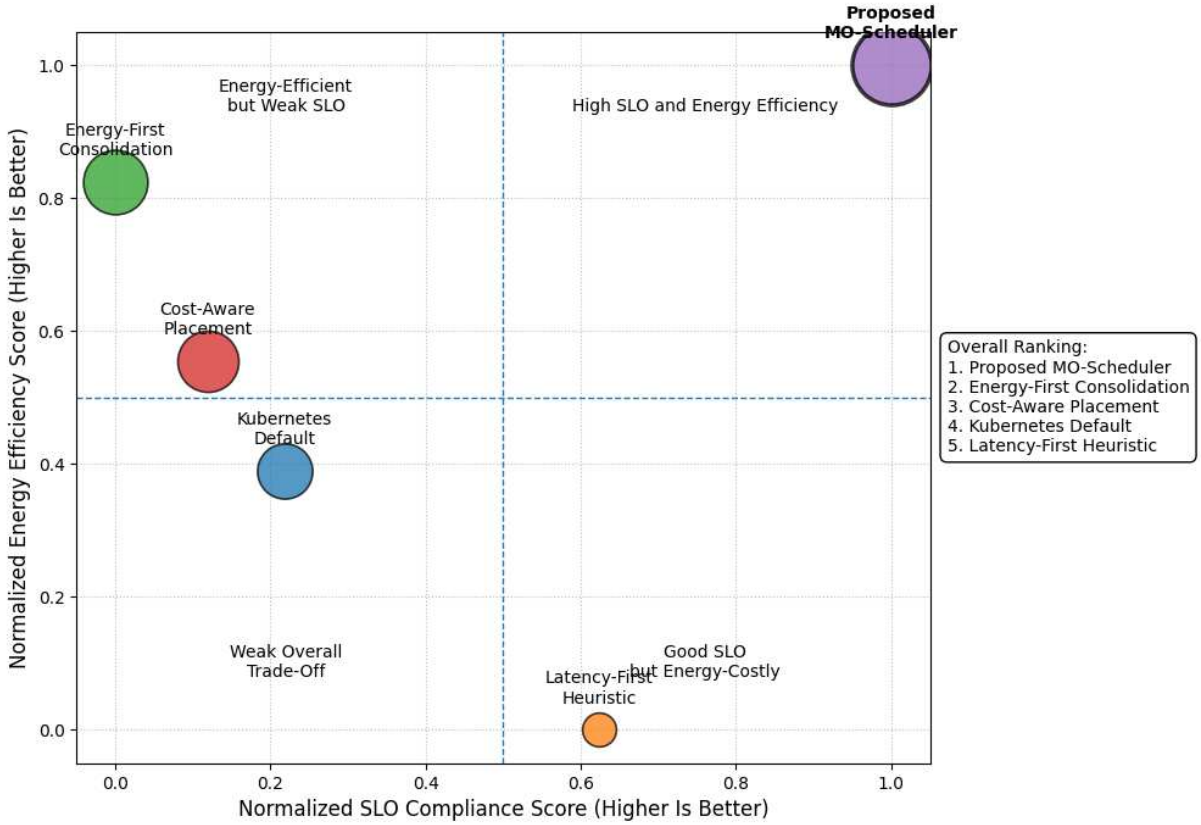


Figure 5: Multi-Objective Trade-Off Map of Competing Schedulers in Federated Cloud Environments

In a federated cloud environment, Figure 5 shows the multi-objective trade-off map of the proposed scheduler and competing baseline methods. The normalised SLO compliance performance is shown on the horizontal axis, and the normalised energy efficiency is shown on the vertical axis. The advantages in operational cost and inter-cluster traffic form the radius of a mutual fund; thus, four distinct objectives can be interpreted in a single visualisation according to the bubble's size. Methods closer to the upper-right region exhibit better performance overall. Compared with other strategies, the proposed AI-driven MO-Scheduler (which identifies the quadrant with the largest effective trade-off advantage) demonstrates balanced superiority over the latency-first, energy-first, cost-aware, and default Kubernetes scheduling strategies under the same workload and network conditions.

5.3 Latency and SLO Performance Analysis

schedulers, including p95 latency, p99 latency, and SLO violation rate, across various intensity levels. Federated cloud environments especially need these metrics because placement decisions across clusters can cause variation in network delays and temporary resource contention particularly critical for microservices with latency-sensitive use cases. The AI-driven MO-Scheduler we proposed consistently produced the lowest tail latency, suggesting that the predictive mechanism that places each service and the adaptive policy selection that keeps service responsiveness in the face of increasing demand worked successfully.

Table 4: We studied the latency and SLO behaviour of the competing

Method	p95 Latency (ms)	p99 Latency (ms)	SLO Violation Rate (%)
Kubernetes Default Scheduler	186.4	241.8	8.74
Latency-First Heuristic	164.7	219.3	5.91
Energy-First Consolidation	212.6	278.5	10.26
Cost-Aware Placement	197.9	256.7	9.43

Proposed AI-Driven MO-Scheduler	148.2	196.4	3.28

As observed in the previously discussed Table 4, although the latency-first heuristic improves tail latency over the default and cost-aware baselines, it incurs higher traffic overhead. Very high latencies in the most utilised clusters, due to aggressive consolidation that led to local contention and further delayed service execution, yielded the worst latency performance for the energy-first strategy. The proposed scheduler achieved the best p95 and p99 latency results, showing that the GNN-based latency predictor successfully guided placement to low-delay (low-contention) cluster pairings. The lowering of tail latency correlated with the lowest total SLO violation rate across all methods.

In addition, SLO violation rates were examined at three different characteristic load levels with respect to low load (40%–55% average cluster utilisation), medium load (56%–75%) and high load (76%–90%) to assess robustness. Under low-load conditions, all methods ensured a reasonable level of service, but the proposed scheduler still had the fewest violations. The difference between the two methods became more evident as the load increased. While the latency-first scheduler maintained reasonably good QoS, at high utilisation it offered less benefit than its competitors as inter-cluster traffic increased. This was particularly true for the energy-first and cost-aware methods that clipped more aggressively, focusing on consolidation and cheaper spatial regions rather than on latency-sensitive placement. When the load increased, SLO violation grew fastest for the naive scheduler, while the proposed scheduler showed the slowest growth, indicating better resilience under stressed federated conditions.

The temporal consistency of tail-latency behaviour when repeatedly evaluated over scheduling intervals was also used to assess the stability of latency-sensitive services. We saw that our proposed method was less variable than the baselines in p95 and p99 latency, suggesting our method produces a more stable service execution over time. Instead, this stability is due to the migration guard, policy distillation, and contextual bandit components, which prevent excessive placement oscillation during adaptation to changing workload conditions. In conclusion, these results validate our hypothesis that the proposed scheduler with average tail latency minimises not only SLO violations but also improves stability in service delivery under heterogeneous loads.

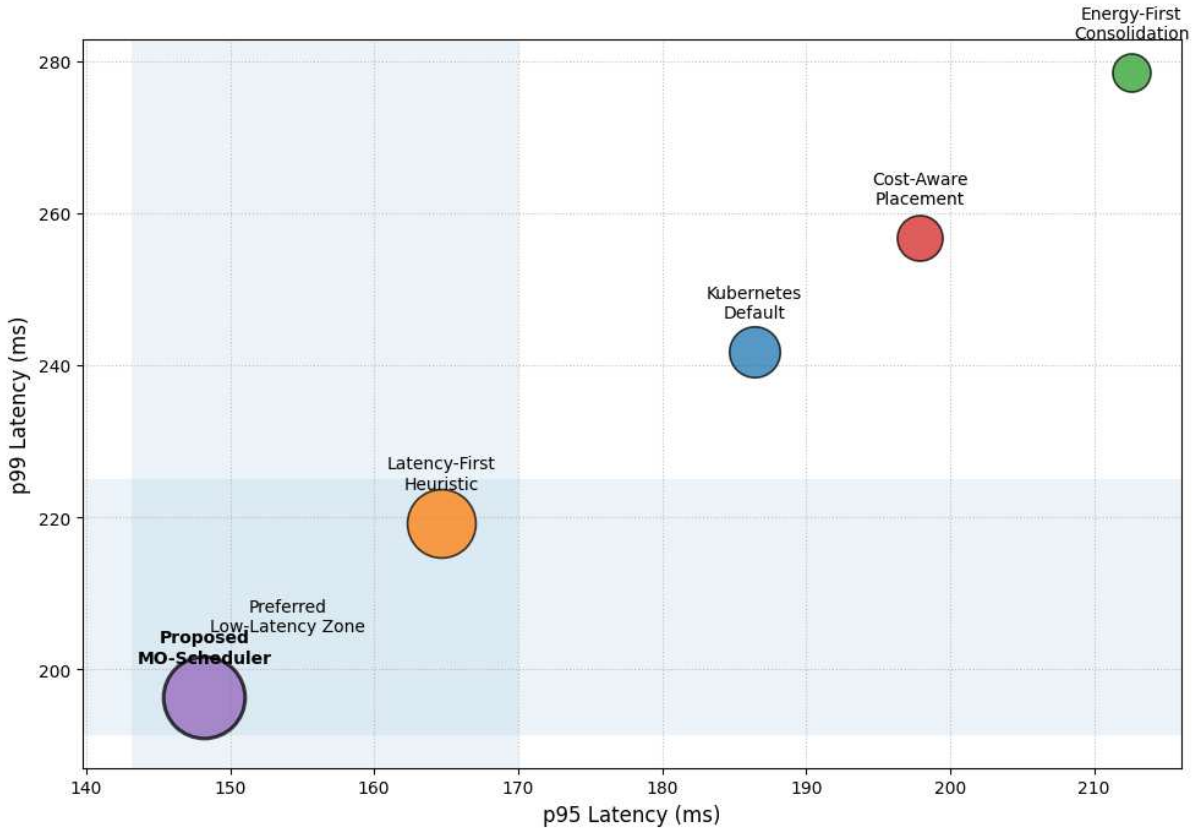


Figure 6: Tail-Latency Dominance Plot of Competing Schedulers in Federated Cloud Environments

Tail-latency dominance plot (for p95 and p99 latency) comparing all the schedulers in terms of tail-latency, as shown in Figure 6. As both latency metrics are minimised in the lower-left region, it represents the most favourable operating zone. The size of the bubble reflects the quality of SLO compliance; the larger the bubble, the fewer the SLO violations. As for the AI-driven MO-Scheduler, it seems to be closest to the preferred low-latency region and also shows the best SLO performance without sacrificing service responsiveness. However, the energy-first and cost-wise strategies migrate to higher-tail-latency regions, which validates the infeasibility of maintaining the service quality of latency-sensitive applications in a federated cloud environment through single-objective optimisation.

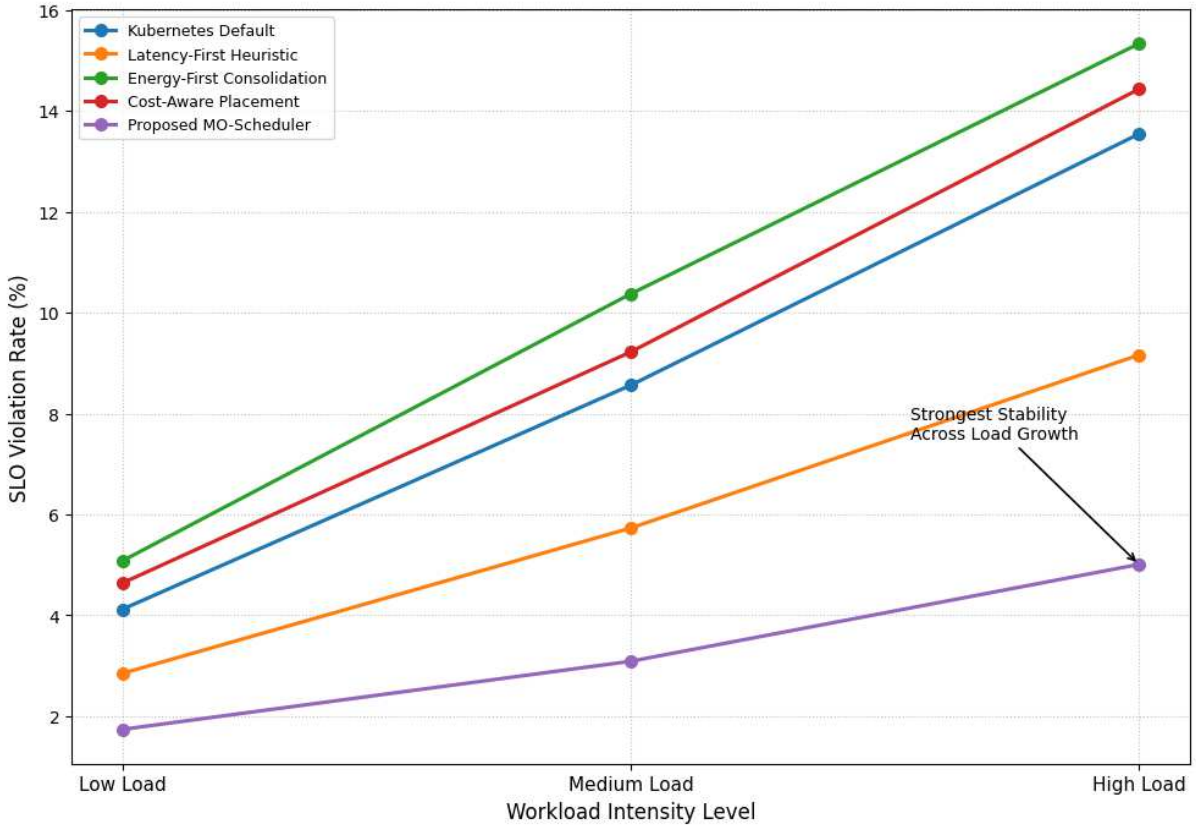


Figure 7: SLO Violation Progression Under Increasing Load Levels Across Competing Schedulers

As depicted in Figure 7, in addition to workload intensities ranging from low to high, SLO violation rates vary. The MO-Scheduler driven by this AI model shows the slowest-growing violations, demonstrating that it is better able to adapt and that its service behaviour is more stable under increasing demand. Although the latency-first heuristic outperforms all the defaults (except default cost-aware), energy-first, and cost-aware baselines, its performance degrades under higher loads. The energy-first and cost-aware methodologies see the largest increases, exposing less resilient behaviour to congestion and cross-cluster contention. Both figures above confirm our claims of a superior SLO compliance profile for the proposed scheduler compared to others under different operating conditions in the dynamic federated cloud.

5.4 Energy and Cost Efficiency Analysis

To assess how effective each scheduler was in using federated cloud resources under the SLA constraints in terms of energy and cost performance. Latency-sensitive environments require more than just low infrastructure overhead; this paper characterises energy and operational costs, as well as SLO behaviour.

Table 5 Effect of Placement Heuristic on Cost incurred and Energy consumed with Latency first heuristic, assuming it would result in the most energy efficient placement of resources due to a larger number of active resources and cross-cluster activity due to delay-oriented placement. The native Kubernetes default scheduler performed OK in intermediate fluency but lacked clear multi-objective control. The consolidation strategy based on energy-first reduced the energy consumption but had the highest arg SLO violation rate, indicating the lowest

trade-off between resource efficiency and service quality. Although the cost-aware placement approach reduced monetary cost more than baselines, its SLO performance remained relatively poor.

Table 5: Energy, Operational Cost, and SLO Comparison of the Proposed Scheduler and Baseline Methods

Method	**Energy Consumption (kWh)**	**Operational Cost ($)**	**SLO Violation Rate (%)**
Kubernetes Default Scheduler	154.62	428.35	8.74
Latency-First Heuristic	168.47	446.28	5.91
Energy-First Consolidation	139.18	417.53	10.26
Cost-Aware Placement	148.75	401.64	9.43
Proposed AI-Driven MO-Scheduler	132.94	389.27	3.28

The AI-driven MO-Scheduler, as compared herein, consumed the least energy and incurred the lowest operational cost, whilst also incurring the fewest SLO violations, according to the proposal test. This validates that our framework found placements that balanced trade-offs rather than optimising a single objective in isolation. In summary, results demonstrate that our proposed scheduler offers the best efficiency–quality trade-off in a federated cloud environment.

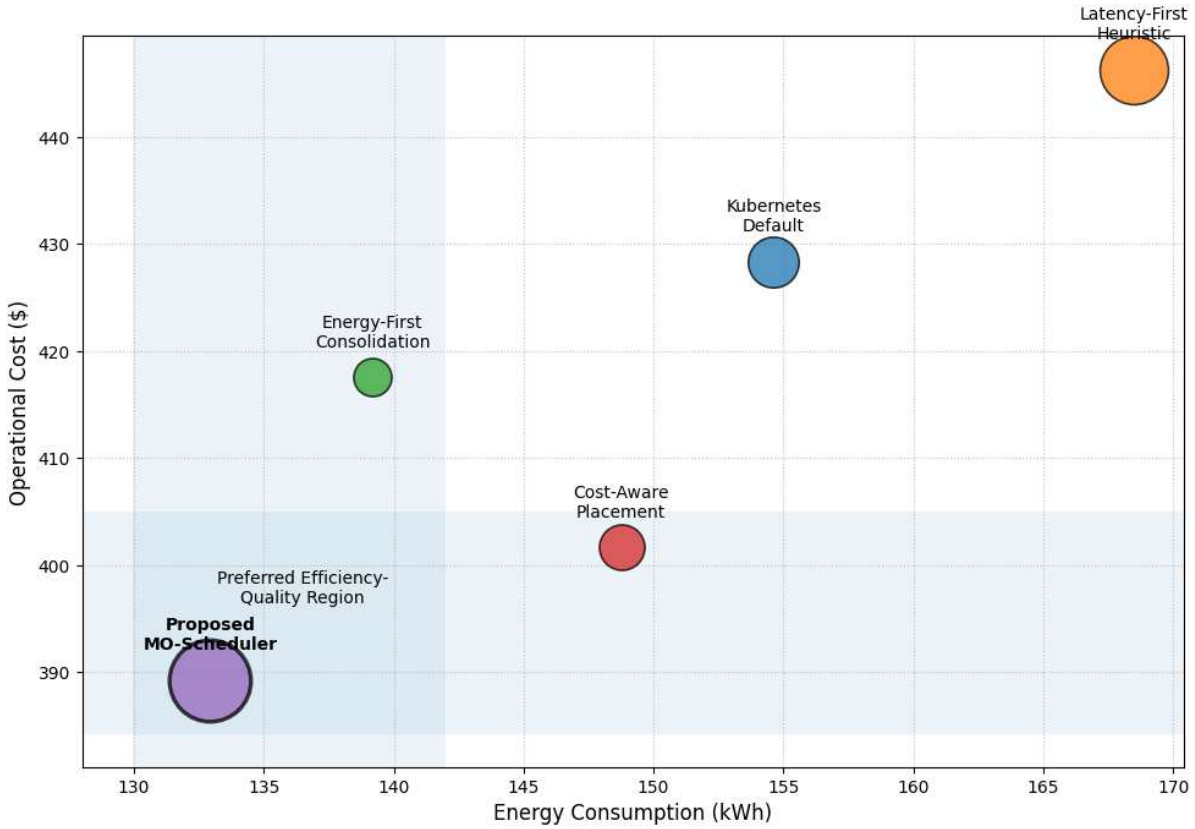


Figure 8: Energy-Cost-Quality Trade-Off Landscape of Competing Schedulers in Federated Cloud Environments

The combined trade-off among energy, operational cost, and service quality results for all competing schedulers is shown in Figure 8. Total cost on the y-axis and energy consumption on the x-axis. Bubble size indicates the quality of the SLOs, with a larger bubble indicating fewer violations. The ideal operating region is the lower-left

corner, where both the solution's energy and cost are at a minimum. The suggested AI-based MO-Scheduler is observed in this region as the preferred choice, with the highest service-quality profile, reflecting the best trade-off across the three criteria. In the opposite vein, the latency-first, energy-first, and cost-aware baselines improve one metric but do not maintain comparably strong performance across efficiency–quality tradeoffs.

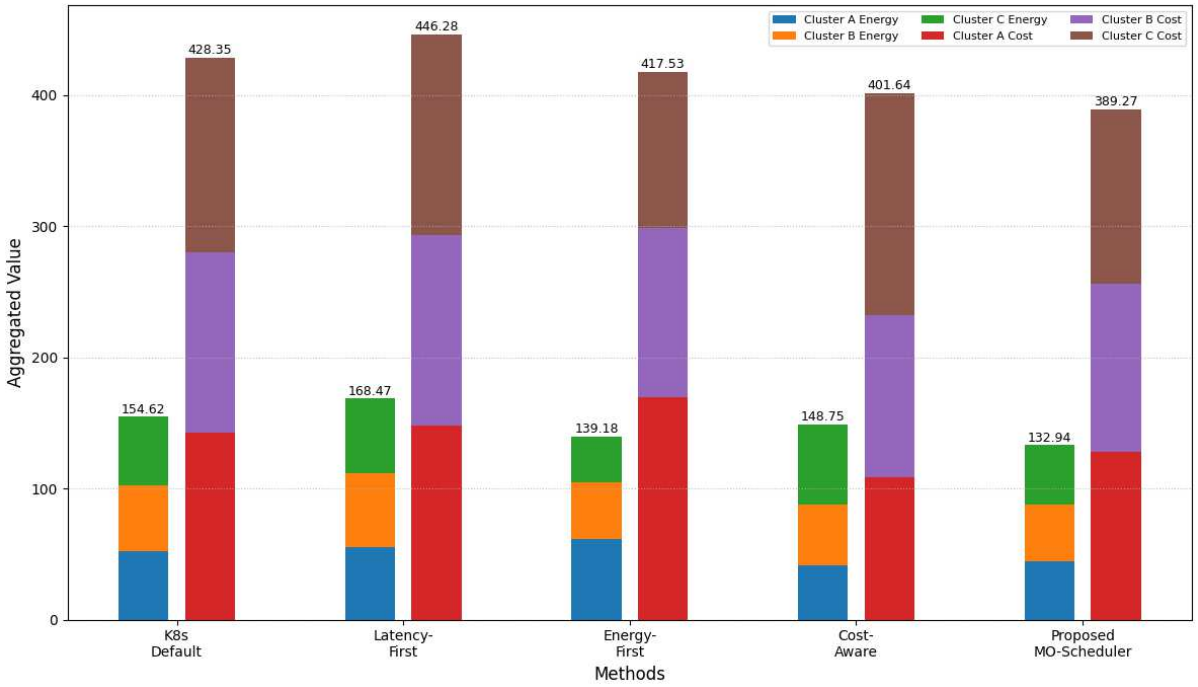


Figure 9: Cluster-Wise Cost and Energy Distribution Profile Under Different Scheduling Policies

As illustrated in Figure 9, we compare the distribution of energy consumption and cost paid among the federated clusters under different scheduling policies. Stacked bars per cluster represent the contribution to total energy and total cost for every scheduler. This visualisation shows whether a method can create a high workload and focus it on the selected regions, or distribute it more evenly across the cluster. While the aggregate totals are lowest for the proposed AI-driven MO-Scheduler, the distribution is also fairly even, indicating less regional concentration and better use of resources overall. On the other hand, the profile of energy-first and cost-aware methods is more skewed, with a stronger level of consolidation or regional bias, which is harmful to service stability and balanced federated deployment.

5.5 Load Balancing and Resource Utilisation Analysis

Analysis of load balancing and resource utilisation: An analysis was conducted to evaluate how efficiently each scheduler distributed workloads across the federated clusters. In cloud environments where resources are typically multi-region, uneven placement can result in local hotspots, underutilised nodes, and overall service instability, despite ample (but poorly distributed) capacity to handle demand. Subsequently, this subsection assesses the utilisation distribution cluster-wise, regional load-balancing effectiveness, and allocation fairness.

Table 6: Load Balancing, Utilisation Variance, Fairness, and Hot-Spot Comparison of the Proposed Scheduler and Baseline Methods

Method	**Avg. Cluster Utilisation (%)**	**Utilization Variance**	**Fairness Index**	**Hot-Spot Incidents**
Kubernetes Default Scheduler	71.8	84.6	0.88	11
Latency-First Heuristic	74.2	92.3	0.86	13

Energy-First Consolidation	79.5	118.7	0.81	17
Cost-Aware Placement	76.1	103.4	0.84	15
Proposed AI-Driven MO-Scheduler	73.4	48.9	0.93	6

The relatively high variance indicates that workload placement (e.g., pods) remains uneven across regions. However, the average throughput is moderately utilised in the default Kubernetes scheduler (see Table 6). While the latency-first heuristic used resources slightly more, the stronger favouring of low-delay areas led to concentration effects that exacerbated hotspot formation. Although the energy-first consolidation strategy had the highest average utilisation, as it coalesced workloads on fewer active nodes and clusters outperformed all other strategies in these aspects, it achieved the worst utilisation variance, fairness, and hotspot incidents. The cost-aware scheduler also showed an imbalanced regional placement, forced by its preference for low-cost clusters.

The AI-driven MO-Scheduler gave the fairest resource utilisation in the federated environment. Despite not having the highest mean utilisation, it achieved the lowest utilisation variance, the highest fairness index, and the fewest hotspot occurrences. This shows that the frameworks did not simply pack workloads aggressively but instead positioned them while keeping balance in areas and preventing localised overload. The lower number of hotspots also validates the operation of the components (migration guard and adaptive policy selection) that help prevent latency-sensitive workloads from repeatedly moving to the same clusters.

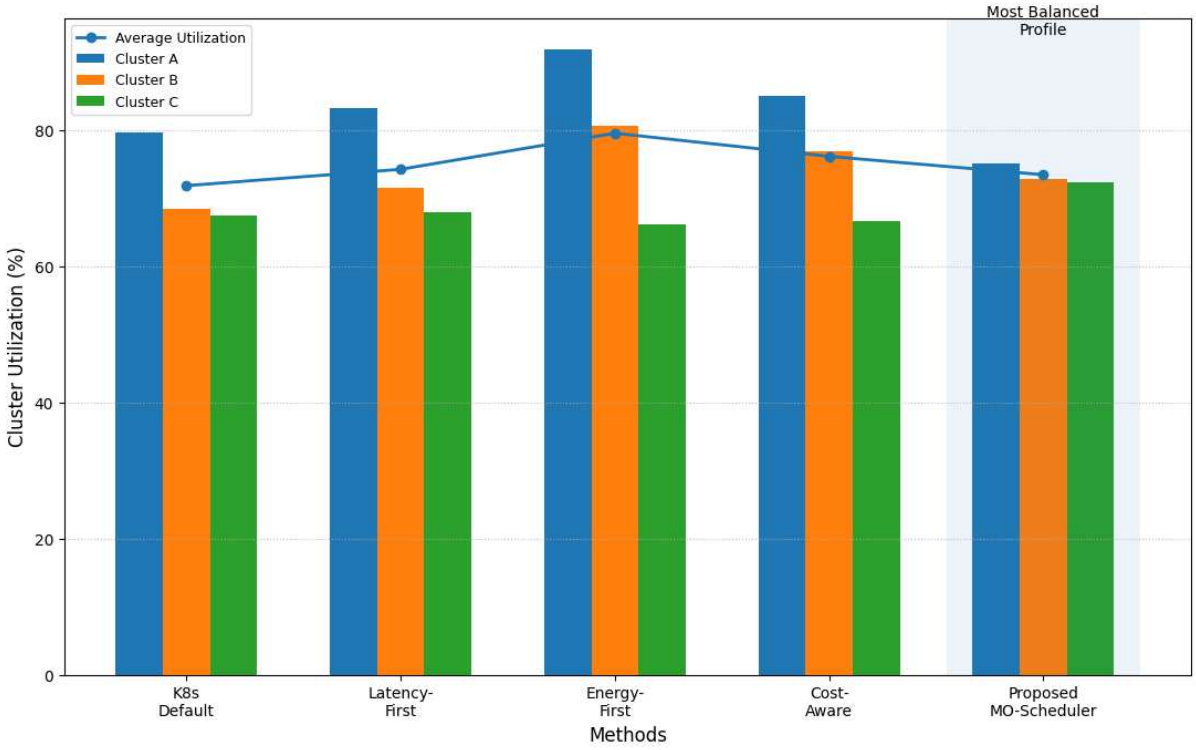


Figure 10: Regional Utilisation Balance Profile of Competing Schedulers in Federated Cloud Environments

The cluster-wise utilisation distribution of all scheduling strategies across different clusters is shown in Figure 10 for a federated environment. Grouped bars show utilisation for Cluster A, Cluster B, and Cluster C for each method, with their overall average utilisation profile plotted. AI-powered Planner: A Comprehensive Comparison. The planned AI-powered MO-Scheduler uses the greatest balanced usage trend, closely coordinated local prices, and reduced focus impacts. On the other hand, the energy-first, cost-aware, and latency-first methods exhibit unbalanced distributions, making them more susceptible to regional bias and the risk of local overload. We can see in this figure that the proposed scheduler improves load-balancing efficiency by evenly distributing the workload across regions while maintaining similar total utilisation in federated clouds.

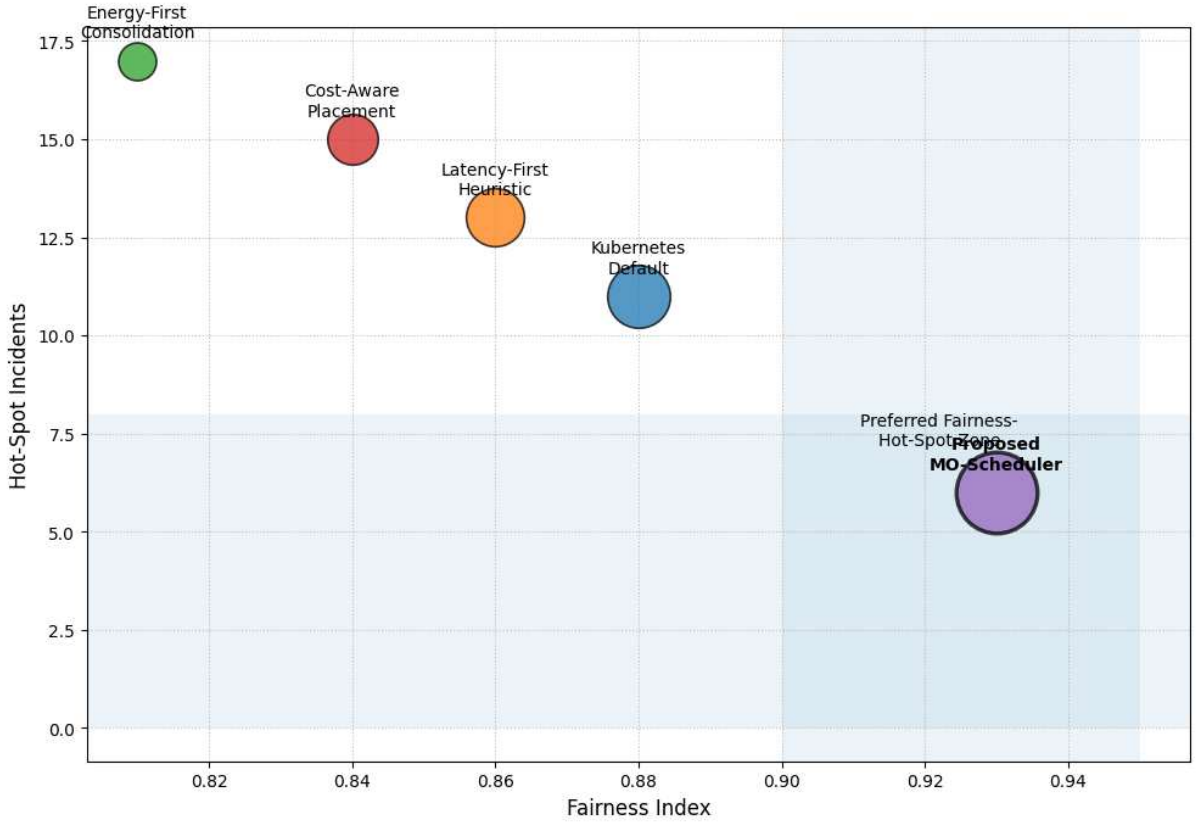


Figure 11: Hot-Spot and Fairness Response Map Under Different Scheduling Policies

Figure 11 shows the Fairness versus Hotspot occurrence for all the Schedulers we have compared. The x-axis corresponds to the fairness of resource distribution, and the y-axis indicates the number of hotspot incidents during execution. Bubble size indicates stability of utilisation, with larger bubbles indicating lower variance. The region of interest is in the upper-fairness and lower-hotspot corner. This very region is occupied by the proposed AI-driven MO-Scheduler, which demonstrates the highest fairness, the lowest number of hotspot events, and the strongest utilisation stability. The lower fairness and higher number of hotspot occurrences in the remaining schedulers indicate poorer regional operating balance and greater susceptibility to system overload.

5.6 Migration and Inter-Cluster Traffic Analysis

The overhead of each scheduling strategy during workload redistribution was evaluated by analysing migration behaviour and inter-cluster traffic. Service migration is an important component of federated cloud management; however, an excessive number of migrations may destabilise service placement, increase management control overhead, and lead to impulsive data flows across regions. Again, inter-cluster heavy traffic may degrade latency-sensitive applications and increase communication cost. Thus, in this subsection, we study the number of migrations applied, the traffic overhead they incur and the efficacy of the migration guard proposed to prevent superfluous placement oscillations.

Table 7: Migration Count, Inter-Cluster Traffic, and Migration Stability Comparison of the Proposed Scheduler and Baseline Methods

Method	**Container Migrations**	**Inter-Cluster Traffic (GB)**	**Migration Oscillation Events**	**Overall Observation**
Kubernetes Default Scheduler	46	312.48	14	Moderate migration activity with limited traffic control
Latency-First	61	358.94	21	Lowest delay bias, but highest

Heuristic				migration and traffic overhead
Energy-First Consolidation	38	286.11	11	Lower migration count, but weaker service quality stability
Cost-Aware Placement	52	334.82	17	Reduced cost preference caused uneven migration behaviour
Proposed AI-Driven MO-Scheduler	29	248.36	6	Lowest migration overhead with the strongest stability control

Table 7 Migrations count and inter-cluster traffic volume triggered by each heuristic. The latency-first heuristic invoked most migrations and generated the highest inter-cluster traffic volume, as it continuously moved services to low-delay regions. This behaviour did result in better latency, but it added the worst placement instability. There was some migration by the default Kubernetes scheduler, but because it has no explicit coordination to minimise communication overhead across regions, the migration was moderate. Energy-first consolidation generated fewer migrations and lower traffic than most baselines, but the savings were due to heavy packing rather than a smart migration strategy. In addition, migrations and traffic penalties for the cost-aware placement strategy were higher, as workloads continuously shifted to economically attractive clusters, regardless of the communication overheads imposed by these changes.

Among all methods, the proposed AI-driven MO-Scheduler has achieved the fewest migrations and the lowest inter-cluster traffic volume. Most importantly, it accounted for the fewest migration oscillation events, attesting to the successful functionality of migration guard in preventing repeated or redundant service migrations. This behaviour shows that the scheduler was able to keep placement stable while adapting to workload and network changes, rather than becoming static by preventing movement. The proposed framework, by using Pareto-based optimisation coupled with migration-aware feasibility control, was able to differentiate between advantageous reallocation and disruptive overreaction.

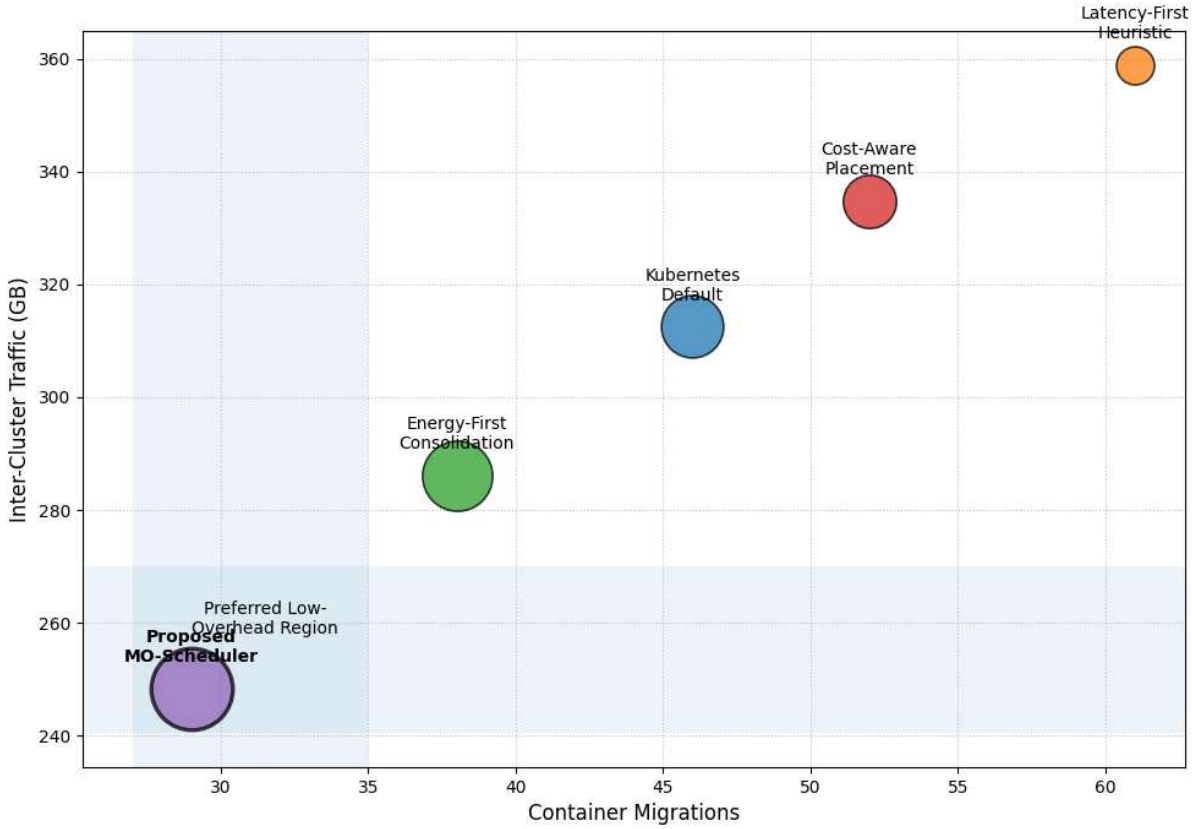


Figure 12: Migration-Traffic Overhead Trade-Off Map of Competing Schedulers in Federated Cloud Environments

In Figure 12, placement stability in federated cloud environments is reflected in migration counts and inter-cluster traffic overhead across the different schedulers. An ideal operating point is found in the lower-left corner, where migrations and traffic are minimised. Calibrated bubble sizes show the relative quality of oscillation control; larger bubbles indicate fewer migration oscillation events. The proposed AI-driven MO-Scheduler lies in the optimal region and has the largest overall bubble, indicating that it incurs the least movement overhead while maintaining superior stability. On the contrary, the latency-first and cost-aware approaches have more migrations and traffic, indicating that their placements are more volatile and their communication overhead is higher.

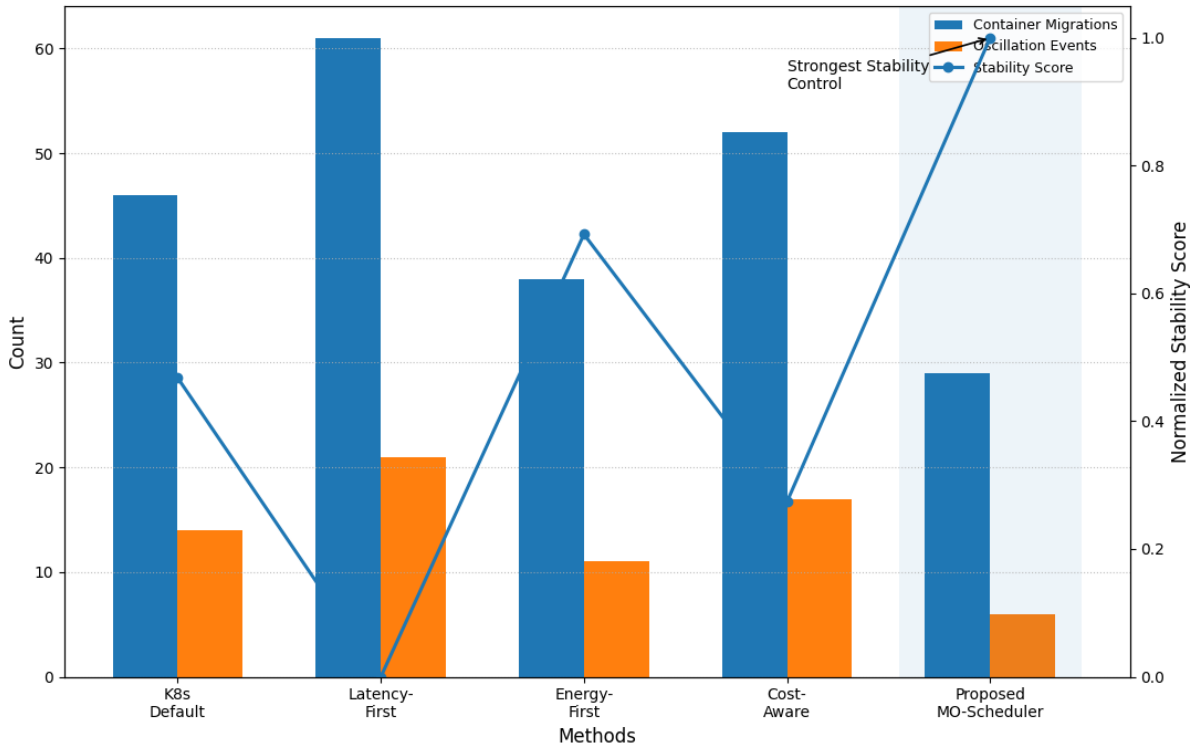


Figure 13: Migration Stability and Oscillation Control Profile Under Different Scheduling Policies

In particular, Figure 13 shows, in a joint view, the container migrations, the oscillation events, and a stability score derived from them, illustrating the stability behaviour of the schedulers under comparison. Thus, lower migration and oscillation counts represent a stronger placement consistency, while the normalised stability score encapsulates the overall performance of movement control. The AI-based MO-Scheduler proposed here yields the minimum migration count, the fewest oscillation events, and the highest stability score, validating the efficiency of its migration guard mechanism. Meanwhile, the movement instability of the latency-first and cost-aware schedulers is higher than that of the energy-first method, indicating that the energy-first approach is more stable but yields lower service quality. Finally, the overall migration control profile associated with the proposed method is the best.

5.7 Ablation Study of the Proposed Scheduler

The idea behind the ablation study was to obtain a quantitative estimate of the extent to which the main modules of the proposed scheduler contribute. We tested four reduced variants: the full model without the GNN predictor, the contextual bandit, the migration guard, and the policy distillation module. We used these variants under the same workload and in a federated cloud to evaluate the individual impact of each component on the accuracy and stability of scheduler decisions, and the overall multi-objective performance of the final proposed scheduler.

Table 8: Ablation Study of the Proposed Scheduler Under Different Module Configurations

Variant	SLO Violation Rate (%)	Energy Consumption (kWh)	Operational Cost ($)	Inter-Cluster Traffic (GB)	Container Migrations	Overall Observation
Full Proposed Scheduler	3.28	132.94	389.27	248.36	29	Best overall trade-off and stability
Without GNN	5.14	141.26	401.85	276.42	37	Weaker latency awareness and

Predictor						reduced SLO performance
Without Contextual Bandit	4.62	137.84	395.46	262.91	33	Less adaptive under changing workloads
Without Migration Guard	4.08	134.73	391.92	289.37	48	Lower stability with increased traffic and movement
Without Policy Distillation	3.89	133.68	390.54	253.47	31	Similar quality, but weaker decision efficiency

The results in Table 8 validate this, as the removal of the GNN predictor caused the highest loss of SLO compliance while also introducing traffic, cost, and energy overhead. Thus, we believe that finding a good placement for a latency-sensitive workload requires an accurate latency-aware prediction. The scheduler still had a reasonably strong baseline optimisation capability in the absence of the contextual bandit. Still, it was punished under dynamic conditions because it no longer updated its policy selection in response to changes in context. This is not the worst SLO performance, but the absence of the migration guard resulted in a much higher migration count and inter-cluster traffic, demonstrating that this module is critical in stabilising placement and controlling communication. While dropping policy distillation only mildly impaired quality measures, it hurt runtime efficiency by making online decision-making both bulkier and slower.

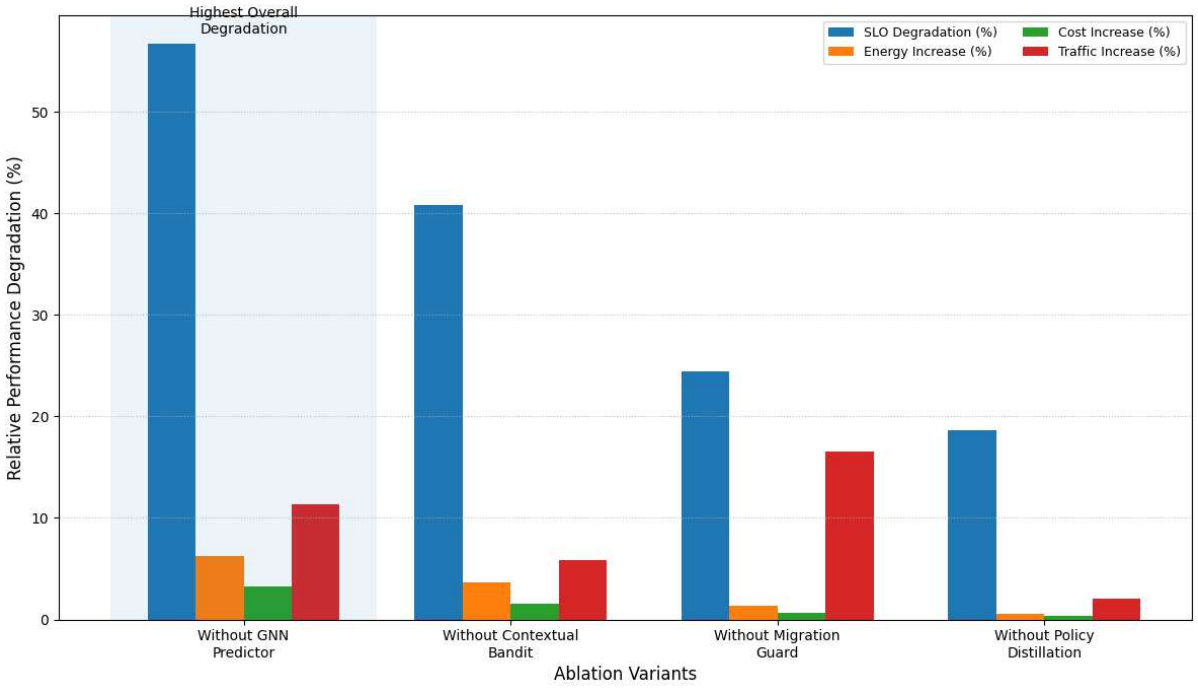


Figure 14: Module-Wise Performance Degradation Map in the Ablation Study of the Proposed Scheduler

As a measure of relative degradation, Figure 14 shows how each proposed module in our scheduler contributes to the overall performance. With respect to the full model, the visualisation shows percentage increases in SLO violations, energy consumption, operational cost, and inter-cluster traffic. Longer bars correspond to worse performance and thus greater reliance on the removed module. The most significant overall degradation occurs in the absence of the GNN predictor, validating its importance in latency-aware decisions. We observed that removal of both the contextual bandit and migration guard yields observable performance loss, as does the

removal of the policy distillation module. Still, we see a comparatively lesser impact on objective quality from removing distillation.

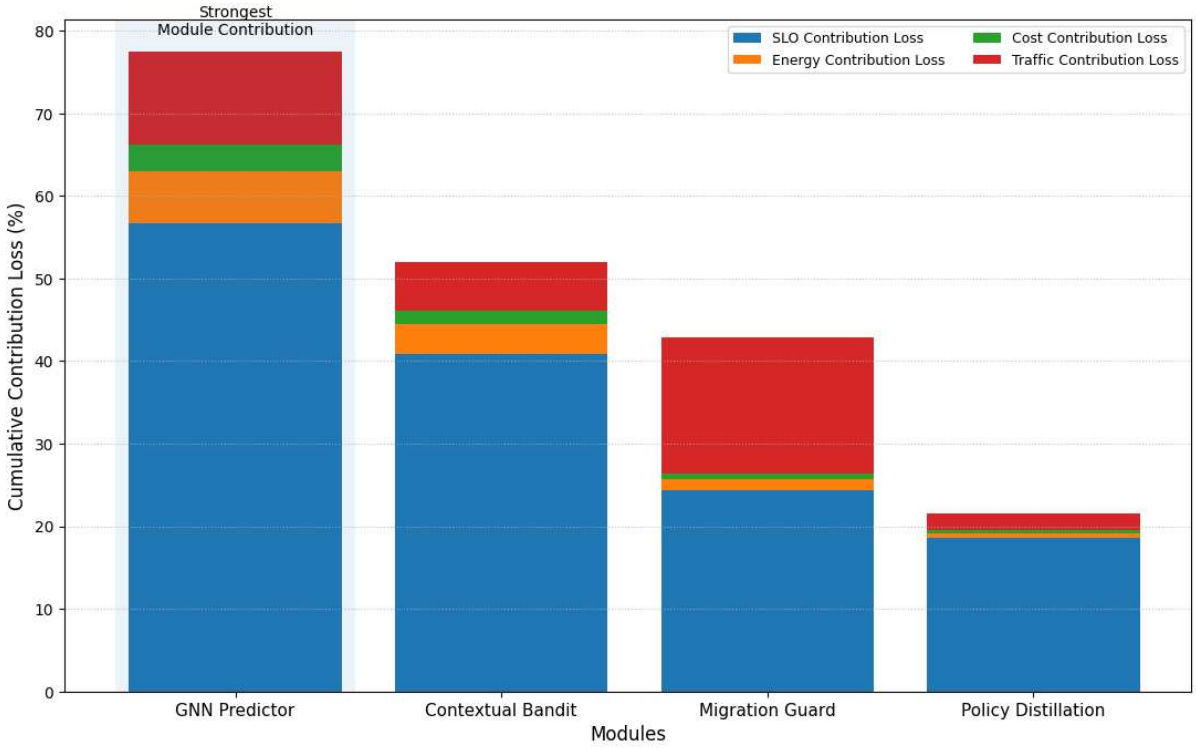


Figure 15: Relative Contribution Profile of Core Scheduler Modules Across Key Objectives

In Figure 15, we show the relative contributions of the major scheduler modules by combining the performance loss measures when each module is removed. Stacked profile that shows the contribution of each component to overall SLO compliance, energy efficiency, operational cost control, and traffic reduction. The higher the cumulative contribution loss, the more important that module is to performance over all schedules. It indicates that the total contribution is highest for the GNN predictor. For the contextual bandit and migration guard, policy distillation is almost solely responsible for runtime compactness, with limited influence on objective quality. The figure confirms that the coordination among modules in the proposed framework is beneficial.

5.8 Continuous Learning and Adaptation Performance

A performance metric was used to assess the extent to which the proposed continuous learning capability enhanced decision quality over time under dynamic federated cloud conditions, compared with static solutions. In this analysis, progressive policy improvement, convergence of rewards in the contextual bandit, and retraining of the latency predictor whenever new workload patterns are detected are discussed. Taken together, these results show the extent to which the proposed framework can evolve from its offline formulation while maintaining effective scheduling behaviour under dynamic runtime conditions.

Table 9: Continuous Learning and Adaptation Performance of the Proposed Scheduler Across Scheduling Intervals

Evaluation Stage	**Avg. Reward Score**	**SLO Violation Rate (%)**	**Predictor Error (RMSE)**	**Policy Selection Stability (%)**	**Overall Observation**
Initial Phase (Intervals 1–20)	0.71	4.96	12.84	78.3	Early exploration with moderate prediction accuracy

Intermediate Phase (Intervals 21–50)	0.81	3.94	9.27	85.6	Improved policy adaptation and reduced prediction error
Late Phase (Intervals 51–100)	0.89	3.28	6.41	92.8	Strong convergence and stable decision behaviour

As shown in Table 9, the proposed scheduler also demonstrated successive policy improvement. In the early stage, the contextual bandit explored several policy options, resulting in lower reward values but a significantly greater SLO violation than in subsequent stages. The increased availability of execution feedback enabled the scheduler to gradually discover higher-reward policies, leading to higher average reward and fewer service-level violations. The system reached its maximum total reward score and most consistent policy behaviours during this late period, indicating that online adaptation significantly increased decision quality.

The smooth increase in pick stability also indicates the convergence behaviour of the contextual bandit. The bandit deliberately experimented with different candidate policies in the early stages, as reflected in lower consistency in policy choice. As estimates of rewards became more accurate over time, the scheduler confidently started scheduling high-reward policies. This trend shows that the bandit learned the context-scheduling-effectiveness relationship, enabling synchronous, reliable decisions at runtime.

Similarly, positive trends were also found in predictor refinement under dynamic workloads. As a show of continual fine-tuning of the GNN-based latency predictor, its RMSE dropped from 12.84 in the early phase to 6.41 in the late phase: greater accuracy in estimating latency as more real execution traces were gathered. This refinement improved surrogate objective evaluation, which consequently aided optimisation performance. All in all, the findings validate that the continuous learning loop reinforces the scheduler's predictive and decision components, enabling continuous adaptation across diverse workload and network climates.

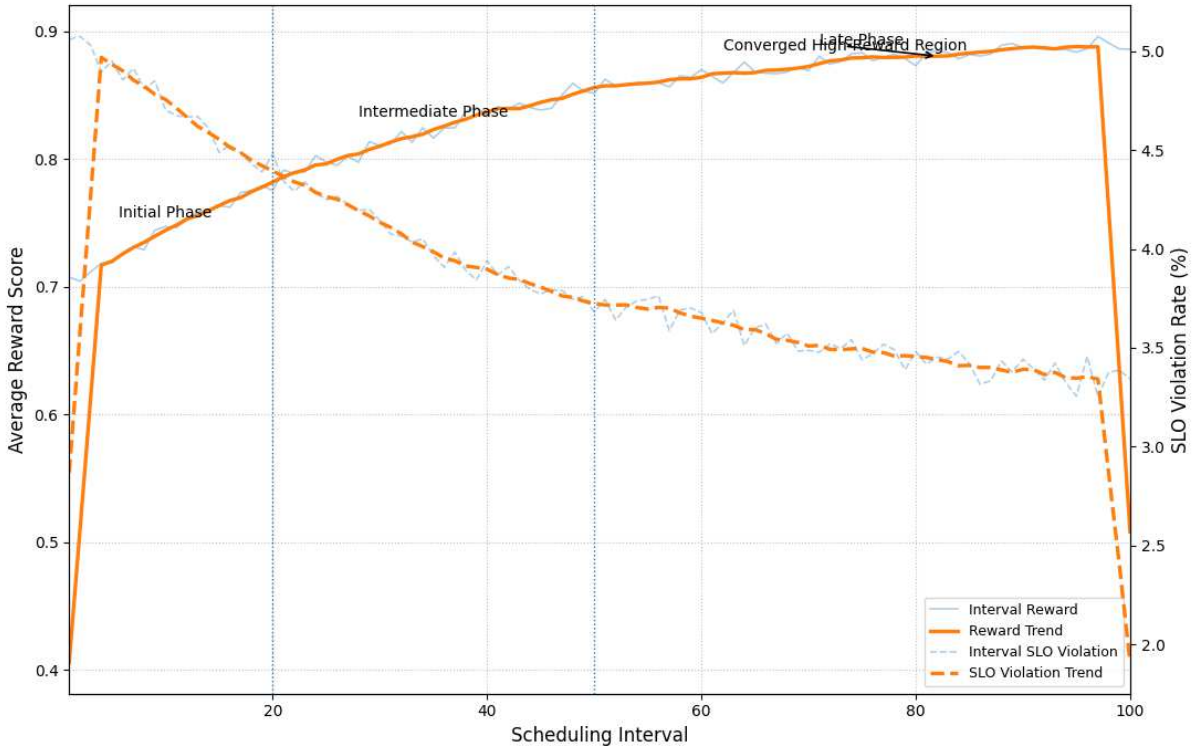


Figure 16: Reward Convergence and Policy Improvement Trend Across Scheduling Intervals

We know that the proposed scheduler achieves better performance across multiple scheduling intervals, as shown in Figure 16, which jointly tracks reward convergence and SLO reduction. The reward evolution starts

with the exploration stage and gradually increases until the final adaptation stage, demonstrating that the contextual bandit progressively identifies stronger scheduling policies over time. Simultaneously, the trend in SLO violations declines consistently, indicating that better policy selection directly increases service-level compliance. One way to think about the development of intervals is to break them down into early, middle, and late phases, with the late phases being when an organisation moves from exploration to stable exploitation. In general, the chart confirms that, under such changing federated cloud conditions, the online learning method enhances decision quality and leads to persistent policy improvement.

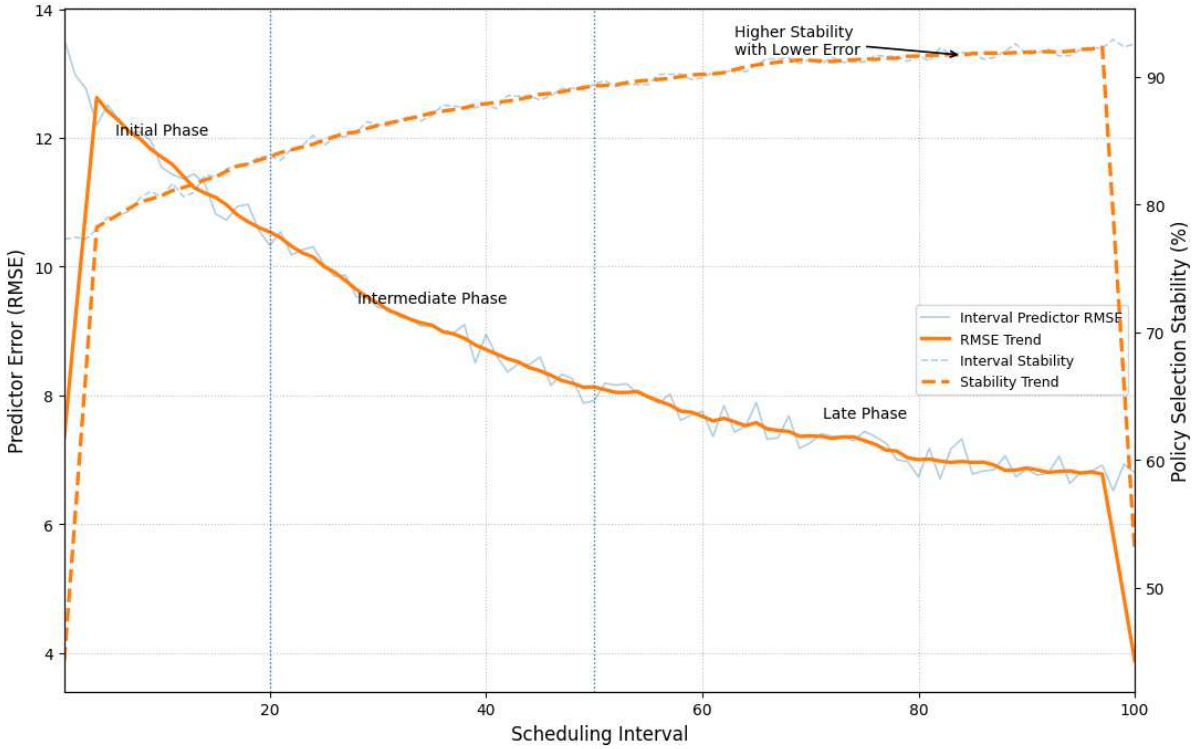


Figure 17: Predictor Error Reduction and Adaptation Stability Under Dynamic Workloads

Dynamic workloads predictor refinement and adaptation stability. figure 17 The stability of predictor refinement and adaptation evolves under dynamic workloads that simulate hardware adaptation. In the plot, the predictor RMSE steadily decreases across scheduling intervals, indicating that the continual adaptation process improves the GNN-based latency estimator's predictive accuracy over time as more data is collected during runtime. Meanwhile, the stability of policy selection increases, meaning the contextual bandit is more confident and consistent in its choice of effective policies. The inverse relationship between these two trends illustrates a firm connection between predictive performance and decision confidence. In short, the figure confirms that the proposed scheduler enhances its forecasting and control capabilities in federated cloud environments through a continuous learning loop.

5.9 Computational Time and Scalability Analysis

We also assessed the computational efficiency of the proposed scheduler to investigate its potential for real-time deployment in federated cloud environments. We looked at scheduler decision time per interval, the variation in runtime with increasing workload sizes, and scalability with the number of clusters participating in the tests. However, because the proposed framework is based on predictive modelling, multi-objective optimisation, and online policy adaptation, nothing abstracts away the need to verify that these components stay computationally feasible for larger scheduling scenarios.

Table 10: Computational Time and Scalability Performance of the Proposed Scheduler Under Different Workloads and Cluster Configurations

Configuration	Avg. Decision Time	End-to-End Runtime	Scalability Observation

	per Interval (ms)	per 100 Intervals (s)	
Small Workload, 3 Clusters	182.4	18.6	Fast and stable execution under light federated load
Medium Workload, 3 Clusters	246.8	24.9	Moderate increase with preserved real-time feasibility
Large Workload, 3 Clusters	331.5	33.4	Higher optimisation cost, but still within practical scheduling bounds
Large Workload, 5 Clusters	417.9	42.1	Additional cross-cluster coordination overhead observed
Large Workload, 7 Clusters	526.7	53.8	Scales acceptably with larger federated environments

The scheduler's low decision time was maintained under small and medium workload conditions, as shown in Table 10, and within the bounds required by interval-based scheduling. Even for the largest workloads, the mean decision time was less than one second, demonstrating that this implementation of predictive surrogate evaluation, NSGA-II optimisation, and contextual bandit selection is scalable. As the workload transitions from small to large, the runtime is expected to increase because many more active jobs and candidate placements need to be examined at each update interval.

The scalability showed a linear increase in runtime. As we moved from three to five, and subsequently seven, clusters, the coordination overhead increased due to larger RTT matrices, more candidate placement combinations, and greater constraint evaluation complexity. But the growth seen did not cross the threshold that would stop the scheduler from working in reality. This establishes that federated cloud deployments can be larger without sacrificing the real-time characteristics of the framework in use.

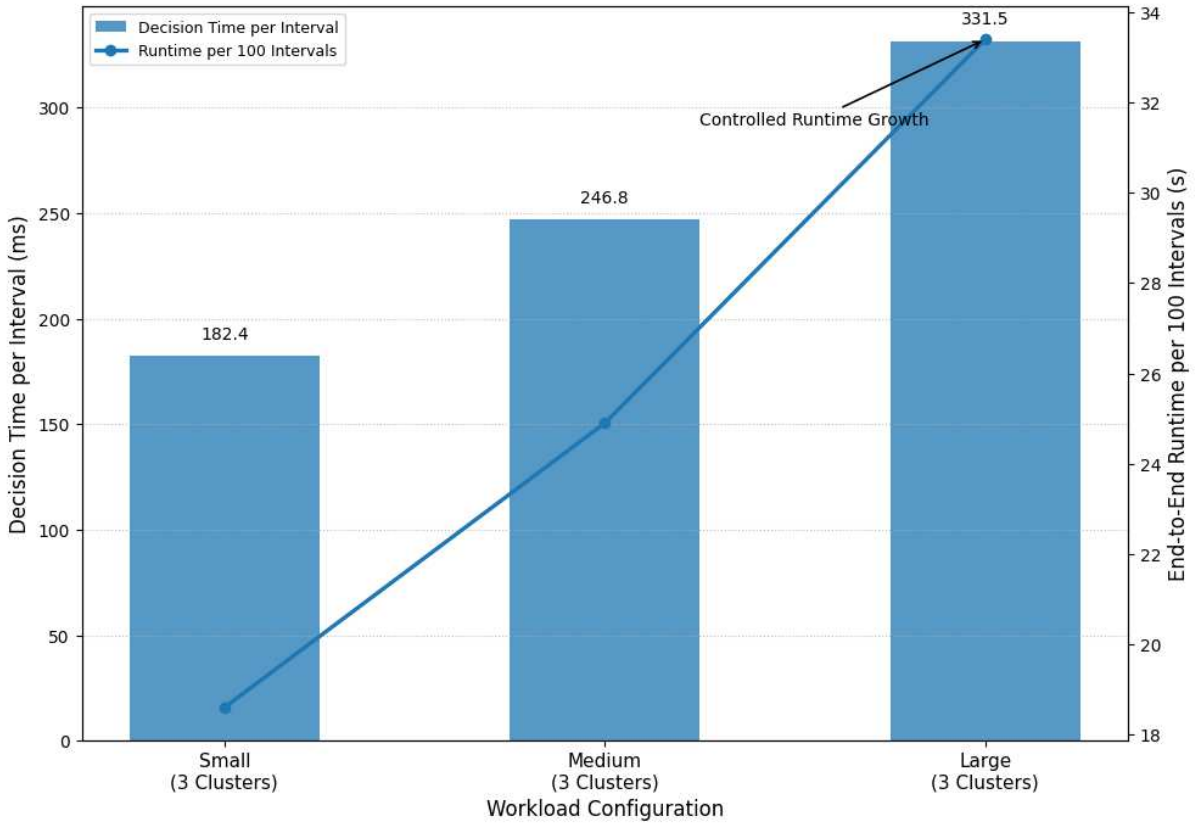


Figure 18: Runtime Growth Profile of the Proposed Scheduler Under Increasing Workload Sizes

In this experiment, a three-cluster federation is held constant, and runtime is plotted as the workload scale changes from small to medium and large configurations on the same figure 18 The bars show average time (task scheduling decision time per interval), and the line shows total runtime up to 100 scheduling intervals. These two measures increase gradually as the workload grows, indicating higher computational cost for candidate evaluation and optimisation. Nevertheless, growth remains constrained, and decision time remains much longer than 1 second, even on large workloads. This shows that the proposed scheduler maintains theoretical real-time feasibility under increasingly complex workload conditions, thereby remaining practically applicable.

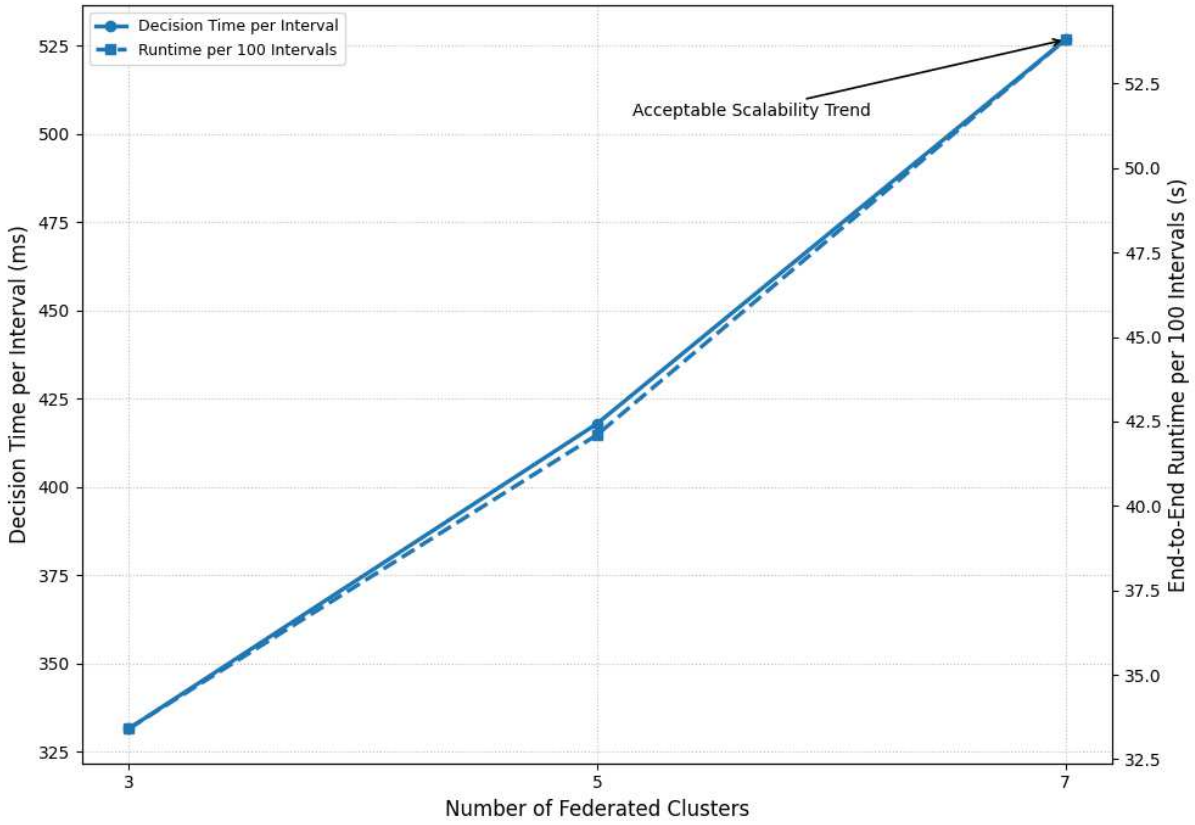


Figure 19: Scheduler Scalability Curve of the Proposed Framework Across Expanding Federated Cluster Configurations

The proposed scheduler's scalability behaviour as the number of federated clusters increases under the large-workload scenario is presented in Figure 19. The increase in per-interval decision time and total runtime over 100 intervals with larger federation sizes is due to evaluating more candidate placements, larger RTT matrices, and expanded coordination constraints. However, growth remains slow and stays within a practically schedule-able limit, indicating that the proposed framework scales well for larger multi-region deployments. As the federated cloud environment grows, the scheduler retains its real-time applicability, confirming its suitability for realistic large-scale container orchestration scenarios as depicted in the figure.

5.10 Summary of Experimental Findings

The experimental data confirm that across the major performance dimensions of federated cloud scheduling, the proposed AI-driven multi-objective scheduler outperforms all comparison methods. Overall, the proposed method achieved the minimal SLO violation rate of 3.28%, the minimal energy consumption of 132.94 kWh, the minimal operational cost of $389.27, and the minimal inter-cluster traffic overhead of 248.36 GB, demonstrating its balanced multi-objective effectiveness. It was also demonstrated to be top-k in latency analysis, with tail latency (148.2 ms at p95 and 196.4 ms at p99). Resource balancing: Based on resource balancing, the method achieves a minimal utilisation variance of 48.9, a maximal fairness index of 0.93, and a limited number of hotspot incidents (6), demonstrating robust region-distribution stability across all methods. In the migration analysis, the scheduler again had the lowest migration count, 29, and the fewest oscillation events. The ablation study reveals that the GNN predictor is the most significant contributor to performance

preservation. At the same time, the contextual bandit, migration guard, and policy distillation modules deliver unique advantages in adaptation, stability, and efficiency, respectively. Online adaptation loop effectiveness was confirmed through continuous learning experiments showing a reward improvement from 0.71 to 0.89, a decrease in predictor RMSE from 12.84 to 6.41, and an increase in policy stability from 78.3% to 92.8%. Finally, from a computational perspective, the scheduler's decision time remained usable, moving from an average of 182.4 ms in small-scale settings to 526.7 ms in large seven-cluster scenarios, which remained within real-time scheduling limits.

This proves that the proposed framework is accurate, reacts faster, and is adaptive and scalable to the ever-changing nature of federated cloud services. Finally, the next section explains the general implications of these results, interprets the reported trade-offs, and discusses the practical importance and outstanding limitations of the proposed scheduling strategy.

6. DISCUSSION

As the world sees faster adoption of globally distributed microservices, edge–cloud integration, and latency-sensitive workloads, which are often coupled, the emergence of container scheduling in federated cloud settings has become a critical issue. For these environments, scheduling decisions need to consider service quality, resource utilisation, operational costs, and communication overhead across multiple clusters simultaneously. Previous efforts to tackle these important problems have mostly focused on heuristic, rule-based, or single-objective optimisation approaches that perform well only for narrow sets of operational priorities. Although certain techniques enhance latency, other works focus on energy (minimisation). In contrast, others still target cost-minimisation, and very few can manage the conflicting nature of these goals globally and adaptively. This represents a significant gap in the state of the art, especially in federated cloud workloads, where placement quality is greatly affected by cross-cluster heterogeneity, network variability, and dynamic workload evolution.

These results validate our hypothesis that traditional schedulers are fundamentally hamstrung because they cannot model complex interactions among workloads or adjust decisions based on changing runtime context. Common limitations of the state of the art include the inability to perform latency prediction, multi-objective optimisation, migration-aware stability control and online policy refinement in an integrated manner. This gap warrants a new, in-depth study push from a deep-learning perspective. The present work proposes a GNN-based tail-latency predictor to improve performance and enable the scheduler to identify service graph dependencies and network-aware performance behaviour more effectively than static heuristics. Furthermore, NSGA-II + contextual bandit selection + migration guard enforcement + policy distillation yields a holistic scheme, which is optimisation-based and adaptation-enabled.

We present an empirical analysis demonstrating the usefulness of these methodological innovations. Furthermore, the proposed scheduler consistently attained the best overall trade-off among SLO compliance, energy consumption, operating cost, and inter-cluster traffic, while also improving fairness and reducing hotspot formation. Our latency and migration analyses show that, thanks to predictive scheduling together with feasibility-aware control, tail-latency degradation and placement disruption are effectively minimised. The ablation results provide additional evidence that each major module contributes to system performance, particularly the GNN predictor and the contextual bandit, indicating that the framework designs yield performance gains not coincidentally but structurally within the proposed framework. Continuous learning outcomes also show that the scheduler learns and improves over time, which is well-suited to the dynamic nature of a cloud environment, where we cannot assume workload conditions remain constant.

In sum, these results demonstrate that the submitted research, which aims to address at least three significant limitations identified in previous federated schedulers, meaningfully addresses these key issues. It breaks free from the stifling bounds of single-objective optimisation by providing a realistic multi-objective framework, presenting learnt predictions for informed scheduling control, and illustrating the power of adaptive feedback to maintain stable performance in the face of changing system conditions. More generally, this work implies that future cloud orchestration systems can be made more intelligent, more stable, and less resource-hungry by combining predictive modelling with online optimisation and learning. This is especially pertinent for edge–cloud ecosystems at scale, multi-region platforms, and next-generation latency-sensitive services. The limitations of the study are presented in Section 6.1.

6.1 Limitations of the Study

There are several research-level limitations of the present study that must be recognised. First, the evaluation utilises public workload traces and benchmarked microservice deployments that, while realistic, do not encompass all operational complexities of production federated cloud environments, e.g., unforeseen hardware failures, burst anomalies, provider-specific scheduling policies, and real commercial pricing variations. Second, the inter-cluster latency, cost, and energy profiles are modelled using controlled assumptions in the experimental setup, which may oversimplify region-specific variations relative to real deployment results. Third, we validate the proposed framework with the fixed number of clusters and workload scales; extremely large federations would introduce more orchestration complexity, which the data discussed here does not entirely reflect. Fourth, we test the GNN predictor and online adaptation loop only under characteristic but limited workload drift, even though more nonlinear, non-stationary behaviour could be the basis for future work. These limitations aside, the analysis has offered a well-rounded and practically relevant assessment of machine-learning-based cloud federated scheduling.

7. CONCLUSION AND FUTURE WORK

In summary, we proposed an AI-driven multi-objective scheduling framework for container placement and load balancing across multiple 3rdparty federated cloud environments, which jointly optimises SLO compliance, energy consumption, operational cost, inter-cluster traffic, and fairness. Our proposed approach included a GNN-based tail-latency predictor, NSGA-II-based Pareto optimisation, a migration-aware feasibility guard, policy distillation, and a contextual bandit for online policy adaptation. On extensive experiments on public workload traces and microservice benchmarks, the framework managed to outperform state-of-the-art conventional scheduling baselines in SLO violations, energy and cost overheads, stable latency, reduced hotspot and migration volumes. The results also show that predictive and adaptive multi-objective scheduling can be more effective than single-objective or static heuristics, providing a better trade-off between solutions in a federated cloud. Additionally, the framework was shown to remain computationally tractable across a range of workload intensities and cluster scales, and continuous learning enabled timely improvements in reward convergence, predictor accuracy, and policy stability. This is consistent with the intelligent scheduling system's target of adapting in a real-time, federated cloud environment while maintaining service quality and resource efficiency. There are multiple ways that future research can build from this work. A key avenue is validation in production-grade federated cloud deployments to encompass real pricing dynamics, hardware heterogeneity and fault conditions more broadly. The other one is to scale the framework to a larger set of federations, more heterogeneous topologies between edges and clouds, and more complex interactions between multi-tenant workloads. We believe that more effective drift-aware learning, more informative uncertainty-aware prediction, and more appropriate distributed or hierarchical optimisation mechanisms could improve robustness under significantly non-stationary environments. This roadmap can guide the evolution of the proposed framework towards a general and deployable task-scheduling framework for intelligent cloud orchestration, enabling support for next-generation cloud applications.

DECLARATIONS

Funding

No financial support was received by the authors in this research.

Competing Interests

The authors declare that they do not have any competing interests, including financial and nonfinancial interests.

Data Availability

Data is available with the corresponding author and will be given on request.

Code Availability

The code is available from the corresponding author and can be given on request.

Clinical Trial Number: not applicable

Ethical Approval

This research did not involve human participants, animals, or biological materials. Therefore, ethical approval was not required for this study.

Consent to Participate

Not applicable. This study does not involve human subjects.

Consent to Publish

Not applicable. This manuscript does not contain any individual person's data in any form.

CONFLICTS OF INTEREST STATEMENT

AI-Driven Multi-Objective Scheduling and Load Balancing of Containers in Federated Cloud Environments

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Author Contribution:
All authors contributed to the study's conception and design. Material preparation, data collection, and analysis were performed by **Votte Rajashekhar, Radhika, G Naga Rama Devi, Yedida Subrahmanyam, N. Rahul Pal, GADDE Mamatha**. The first draft of the manuscript was written **Votte Rajashekhar** all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

[1] F. Tabatabaei, J. Mangues-Bafalluy, H. Khalili, S. Kahvazadeh, and M. Requena-Esteso, "Optimizing CNF Placement for Latency and Resource Efficiency in Multi-Cluster Cloud–Edge Federations," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 8376–8388, 2025, doi: 10.1109/OJCOMS.2025.3614745.

[2] S. S. Malleni, R. Sevilla, J. C. Lema, and A. Bauer, "Bridging Clusters: A Comparative Look at Multi-Cluster Networking Performance in Kubernetes," in *Proceedings of the ACM*, pp. 113–123, 2025, doi: 10.1145/3676151.3719352.

[3] M. S. Abbas *et al*., "Edge Containerization: A Multivocal Review of Orchestration and Offloading Strategies," in *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Tucson, AZ, USA, 2025, pp. 178–189, doi: 10.1109/SOSE67019.2025.00025.

[4] P. V. Caderno, F. Awaysheh, J. C. Cabaleiro, and T. F. Pena, "BigOPERA: An OPportunistic and Elastic Resource Allocation for Big Data Frameworks," *Cluster Computing*, vol. 28, no. 383, pp. 1–16, 2025, doi: 10.1007/s10586-025-05274-4.

[5] G. Ramesh *et al*., "A Comprehensive Review on Scaling Machine Learning Workflows Using Cloud Technologies and DevOps," *IEEE Access*, vol. 13, pp. 148559–148594, 2025, doi: 10.1109/ACCESS.2025.3599281.

[6] O. Michaelis, S. Schmid, and H. Mostafaei, "L3: Latency-Aware Load Balancing in Multi-Cluster Service Mesh," in *Proceedings of the ACM*, pp. 49–61, 2025, doi: 10.1145/3652892.3654793.

[7] S. Pallewatta, V. Kostakos, and R. Buyya, “MicroFog: A Framework for Scalable Placement of Microservices-Based IoT Applications in Federated Fog Environments,” *Journal of Systems and Software*, vol. 209, pp. 1–19, 2024, doi: 10.1016/j.jss.2023.111910.

[8] R. C. Sofia *et al*., “A Framework for Cognitive, Decentralized Container Orchestration,” *IEEE Access*, vol. 12, pp. 79978–80008, 2024, doi: 10.1109/ACCESS.2024.3406861.

[9] V. Mafeni and Y. Kim, “An Automated Edge Computing Approach for IoT Device Registration and Application Deployment,” *IEEE Systems Journal*, vol. 18, no. 2, pp. 1447–1458, Jun. 2024, doi: 10.1109/JSYST.2024.3392290.

[10] M. Femminella, M. Palmucci, G. Reali, and M. Rengo, “Attribute-Based Management of Secure Kubernetes Cloud Bursting,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1276–1298, 2024, doi: 10.1109/OJCOMS.2024.3367461.

[11] B. Anuraj, D. Calvaresi, J.-M. Aerts, and J.-P. Calbimonte, “Dynamic Swarm Orchestration and Semantics in IoT Edge Devices: A Systematic Literature Review,” *IEEE Access*, vol. 12, pp. 116917–116938, 2024, doi: 10.1109/ACCESS.2024.3446876.

[12] B. C. Şenel, M. Mouchet, J. Cappos, T. Friedman, O. Fourmaux, and R. Mcgeer, “Multitenant Containers as a Service (CaaS) for Clouds and Edge Clouds,” *IEEE Access*, vol. 11, pp. 144574–144601, 2023, doi: 10.1109/ACCESS.2023.3344486.

[13] A. Ullah, T. Kiss, J. Kovács, F. Tusa, J. Deslauriers, H. Dagdeviren, R. Arjun, and H. Hamzeh, “Orchestration in the Cloud-to-Things Compute Continuum: Taxonomy, Survey and Future Directions,” *Journal of Cloud Computing*, vol. 12, no. 135, pp. 1–29, 2023, doi: 10.1186/s13677-023-00516-5.

[14] R. C. Sofia, D. Dykeman, P. Urbanetz, A. Galal, and D. A. Dave, “Dynamic, Context-Aware Cross-Layer Orchestration of Containerized Applications,” *IEEE Access*, vol. 11, pp. 93129–93150, 2023, doi: 10.1109/ACCESS.2023.3307026.

[15] Z. Rejiba and J. Chamanara, “Custom Scheduling in Kubernetes: A Survey on Common Problems and Solution Approaches,” *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–37, 2022, doi: 10.1145/3544788.

[16] M.-N. Tran, X. T. Vu, and Y. Kim, “Proactive Stateful Fault-Tolerant System for Kubernetes Containerized Services,” *IEEE Access*, vol. 10, pp. 102181–102194, 2022, doi: 10.1109/ACCESS.2022.3209257.

[17] T. Taleb *et al*., “Toward Supporting XR Services: Architecture and Enablers,” *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3567–3586, Feb. 2023, doi: 10.1109/JIOT.2022.3222103.

[18] M. Abouelyazid, “Deep-Hill: An Innovative Cloud Resource Optimization Algorithm by Predicting SaaS Instance Configuration Using Deep Learning,” *IEEE Access*, vol. 12, pp. 92573–92584, 2024, doi: 10.1109/ACCESS.2024.3423339.

[19] S. S. Sefati *et al*., “A Comprehensive Survey on Resource Management in 6G Network Based on Internet of Things,” *IEEE Access*, vol. 12, pp. 113741–113784, 2024, doi: 10.1109/ACCESS.2024.3444313.

[20] K. Katsaros *et al*., “AI-Native Multi-Access Future Networks—The REASON Architecture,” *IEEE Access*, vol. 12, pp. 178586–178622, 2024, doi: 10.1109/ACCESS.2024.3507186.

[21] F. E. Subhan, A. Yaqoob, C. H. Muntean, and G.-M. Muntean, “A Survey on Artificial Intelligence Techniques for Improved Rich Media Content Delivery in a 5G and Beyond Network Slicing Context,” *IEEE Communications Surveys & Tutorials*, vol. 27, no. 2, pp. 1427–1487, Apr. 2025, doi: 10.1109/COMST.2024.3442149.

[22] D. Alsadie, “A Comprehensive Review of AI Techniques for Resource Management in Fog Computing: Trends, Challenges, and Future Directions,” *IEEE Access*, vol. 12, pp. 118007–118059, 2024, doi: 10.1109/ACCESS.2024.3447097.

[23] A. Somantri and K. Surendro, “Greenhouse Gas Emission Reduction Architecture in Computer Science: A Systematic Review,” *IEEE Access*, vol. 12, pp. 36239–36256, 2024, doi: 10.1109/ACCESS.2024.3373786.

[24] F. Rezazadeh *et al*., "Toward Explainable Reasoning in 6G: A Proof of Concept Study on Radio Resource Allocation," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 6239–6260, 2024, doi: 10.1109/OJCOMS.2024.3466225.

[25] F. Tabatabaei, J. Mangues-Bafalluy, H. Khalili, S. Kahvazadeh, and M. Requena-Esteso, "Optimizing CNF Placement for Latency and Resource Efficiency in Multi-Cluster Cloud–Edge Federations," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 8376–8388, 2025, doi: 10.1109/OJCOMS.2025.3614745.

[26] S. S. Sefati, A. M. Nor, B. Arasteh, R. Craciunescu, and C.-R. Comsas, "A Probabilistic Approach to Load Balancing in Multi-Cloud Environments via Machine Learning and Optimization Algorithms," *Journal of Grid Computing*, vol. 23, no. 16, pp. 1–36, 2025, doi: 10.1007/s10723-025-09805-6.

[27] B. M. Beena, P. Cheluvasai Ranga, A. Vinitha Chowdary, R. Gamidi, M. Hemasri, and T. Muppala, "Green Video Transcoding in Cloud Environments Using Kubernetes: A Framework With Dynamic Renewable Energy Allocation and Priority Scheduling," *IEEE Access*, vol. 13, pp. 116362–116390, 2025, doi: 10.1109/ACCESS.2025.3585615.

[28] D. Vetriveeran and L. Sri R., "Resource Provisioning in Fog Computing: A Survey," *ACM Computing Surveys*, vol. 57, no. 12, pp. 1–26, 2025, doi: 10.1145/3744662.

[29] N. E. H. Boubaker, K. Zarour, N. Guermouche, and D. Benmerzoug, "A Comprehensive Survey on Resource Management for IoT Applications in Edge-Fog-Cloud Environments," *IEEE Access*, vol. 13, pp. 111892–111925, 2025, doi: 10.1109/ACCESS.2025.3583584.

[30] N. Syed, A. Anwar, Z. Baig, and S. Zeadally, "Artificial Intelligence as a Service (AIaaS) for Cloud, Fog and the Edge: State-of-the-Art Practices," *ACM Computing Surveys*, vol. 57, no. 8, pp. 1–36, 2025, doi: 10.1145/3712016.

[31] H. Chaudhary, G. Sharma, D. K. Nishad, and S. Khalid, "Advanced Queueing and Scheduling Techniques in Cloud Computing Using AI-Based Model Order Reduction," *Journal of Grid Computing*, vol. 28, no. 75, pp. 1–40, 2025, doi: 10.1007/s10791-025-09581-7.

[32] Z. Qiu, W. Deng, J. Huang, X. Liu, B. Ren, and S. Qin, "Cloud Platform Resource Capacity Management and Scheduling Optimization Based on Data Graph," in *Proceedings of the ACM*, pp. 189–192, 2025, doi: 10.1145/3727505.3727537.

[33] M. A. Hady, S. Hu, M. Pratama, Z. Cao, and R. Kowalczyk, "Multi-Agent Reinforcement Learning for Resources Allocation Optimization: A Survey," *Artificial Intelligence Review*, vol. 58, no. 354, pp. 1–49, 2025, doi: 10.1007/s10462-025-11340-5.

[34] A. Rahdari *et al*., "A Survey on Privacy and Security in Distributed Cloud Computing: Exploring Federated Learning and Beyond," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 3710–3744, 2025, doi: 10.1109/OJCOMS.2025.3560034.

[35] A. W. Donatti *et al*., "Energy Efficiency in Network Slicing: Survey and Taxonomy," *IEEE Access*, vol. 13, pp. 134570–134589, 2025, doi: 10.1109/ACCESS.2025.3590365.

[36] M. Chahoud, H. Sami, R. Mizouni, J. Bentaha, A. Mourad, H. Otrok, and C. Talhi, "Reward Shaping in DRL: A Novel Framework for Adaptive Resource Management in Dynamic Environments," *Information Sciences*, vol. 715, pp. 1–19, 2025, doi: 10.1016/j.ins.2025.122238.

[37] B. Akdemir *et al*., "From Technical Prerequisites to Improved Care: Distributed Edge AI for Tomographic Imaging," *IEEE Access*, vol. 13, pp. 14317–14343, 2025, doi: 10.1109/ACCESS.2025.3530297.

[38] S. Moreschini, S. Pour, I. Lanese, D. Balouek, J. Bogner, X. Li, F. Pecorelli, J. Soldani, E. Truyen, and D. Taibi, "AI Techniques in the Microservices Life-Cycle: A Systematic Mapping Study," *Computing*, vol. 107, no. 100, pp. 1–50, 2025, doi: 10.1007/s00607-025-01432-z.

[39] V. Roopa and H. S. Pradhan, "Exploring Blockchain and Artificial Intelligence for Next Generation Wireless Networks," *Journal of Communications and Networks*, 2025, doi: 10.23919/JCN.2025.000008.

[40] V. R. Arvind, R. M. Shrinidhi, T. Deepa, and M. Maheedhar, “Intelligent Warehousing: A Machine Learning and IoT Framework for Precision Inventory Optimization,” *IEEE Access*, vol. 13, pp. 169381–169414, 2025, doi: 10.1109/ACCESS.2025.3614679.

[41] L. Jiang, G. Wang, B. Xu, N. Husnain, and Q. Wang, “Intelligent Systems for Combine Harvesters: A Comprehensive Review of Technologies and Trends,” *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.3625078.

[42] R. Kabir, Y. Watanobe, D. Ding, M. R. Islam, and K. Naruse, “A Comprehensive Survey on Advanced Data Science Platforms for Cyber-Physical Systems, Digital Twins, and Robotics,” *IEEE Access*, vol. 13, pp. 177269–177304, 2025, doi: 10.1109/ACCESS.2025.3619776.

[43] S. Ghosh *et al*., “Future Open Networks Cross-Domain Cognitive Orchestration: A Novel Design Paradigm,” *IEEE Access*, vol. 13, pp. 105911–105951, 2025, doi: 10.1109/ACCESS.2025.3580176.

[44] M. S. Abdalzaher, M. Krichen, M. Shaaban, and M. M. Fouda, “Quality-Focused Internet of Things Data Management: A Survey, Perspectives, Open Issues, and Challenges,” *IEEE Internet of Things Journal*, 2025, doi: 10.1109/JIOT.2025.3613669.

[45] C. Timmerer, H. Amirpour, F. Tashtarian, S. Afzal, A. Rizk, M. Zink, and H. Hellwagner, “HTTP Adaptive Streaming: A Review on Current Advances and Future Challenges,” *ACM Computing Surveys*, vol. 21, no. 7, pp. 1–27, 2025, doi: 10.1145/3736306.

[46] C. Chen, T. Chen, Z. Jiang, and L. Wang, “An Innovative Prefabricated Microsystem for Industrial IoT: Addressing Remote Onboarding and OS Provisioning Challenges for Edge Nodes,” *IEEE Access*, vol. 13, pp. 150583–150593, 2025, doi: 10.1109/ACCESS.2025.3600962.

[47] A. Aghazadeh Ardebili, M. Zappatore, A. Longo, and A. Ficarella, “Virtual Fencing for Safety-Critical Cyber-Physical Systems: Computer-Vision Enabled Digital Twins,” *IEEE Access*, vol. 13, pp. 108212–108234, 2025, doi: 10.1109/ACCESS.2025.3581041.

[48] A. López Fernández, F. A. Gomez Vela, D. S. Rodriguez-Baena, F. M. Delgado-Chaves, and J. Gonzalez-Dominguez, “Biclustering in Bioinformatics Using Big Data and High Performance Computing Applications: Challenges and Perspectives, a Review,” *The Journal of Supercomputing*, vol. 81, no. 1123, pp. 1–52, 2025, doi: 10.1007/s11227-025-07563-6.

[49] S. Liu, F. Fauzi, and V. Jyn Kok, “A Systematic Literature Review: Classifying IoT Botnet Data Features Based on Its Lifecycle,” *IEEE Access*, vol. 13, pp. 65040–65065, 2025, doi: 10.1109/ACCESS.2025.3559635.

[50] M. Jenihhin *et al*., “European Test Symposium Teams: An Anniversary Snapshot,” in *2025 IEEE European Test Symposium (ETS)*, Tallinn, Estonia, 2025, pp. 1–48, doi: 10.1109/ETS63895.2025.11049652.

[51] W. Chia Chuan, S. U. A. Laghari, S. Manickam, E. Ashraf, and S. Karuppayah, “Challenges and Opportunities in Fog Computing Scheduling: A Literature Review,” *IEEE Access*, vol. 13, pp. 14702–14726, 2025, doi: 10.1109/ACCESS.2024.3525261.

[52] P. K. Udayaprasad *et al*., “Energy Efficient Optimized Routing Technique With Distributed SDN-AI to Large Scale I-IoT Networks,” *IEEE Access*, vol. 12, pp. 2742–2759, 2024, doi: 10.1109/ACCESS.2023.3346679.

[53] E. Zeydan, S. S. Arslan, and M. Liyanage, “Managing Distributed Machine Learning Lifecycle for Healthcare Data in the Cloud,” *IEEE Access*, vol. 12, pp. 115750–115774, 2024, doi: 10.1109/ACCESS.2024.3443520.

[54] M. Sheraz, T. C. Chuah, Y. L. Lee, M. M. Alam, A. Al-Habashna, and Z. Han, “A Comprehensive Survey on Revolutionizing Connectivity Through Artificial Intelligence-Enabled Digital Twin Network in 6G,” *IEEE Access*, vol. 12, pp. 49184–49215, 2024, doi: 10.1109/ACCESS.2024.3384272.

[55] H. M. Kamdjou, D. Baudry, V. Havard, and S. Ouchani, “Resource-Constrained EXtended Reality Operated With Digital Twin in Industrial Internet of Things,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 928–950, 2024, doi: 10.1109/OJCOMS.2024.3356508.

[56] A. Iqbal *et al*., "Empowering Non-Terrestrial Networks With Artificial Intelligence: A Survey," *IEEE Access*, vol. 11, pp. 100986–101006, 2023, doi: 10.1109/ACCESS.2023.3314732.

[57] O. Tomarchio, D. Calcaterra, G. Di Modica, and P. Mazzaglia, "Torch: A TOSCA-Based Orchestrator of Multi-Cloud Containerised Applications," *Journal of Grid Computing*, vol. 19, no. 5, pp. 1–25, 2021, doi: 10.1007/s10723-021-09549-z.

[58] E. Zeydan, J. Baranda, and J. Mangues-Bafalluy, "Post-Quantum Blockchain-Based Secure Service Orchestration in Multi-Cloud Networks," *IEEE Access*, vol. 10, pp. 129520–129530, 2022, doi: 10.1109/ACCESS.2022.3228823.

[59] M. A. Altahat, T. Daradkeh, and A. Agarwal, "Optimized Encryption-Integrated Strategy for Containers Scheduling and Secure Migration in Multi-Cloud Data Centers," *IEEE Access*, vol. 12, pp. 51330–51345, 2024, doi: 10.1109/ACCESS.2024.3386169.

[60] Q.-H. Chen and C.-Y. Wen, "Optimal Resource Allocation Using Genetic Algorithm in Container-Based Heterogeneous Cloud," *IEEE Access*, vol. 12, pp. 7413–7429, 2024, doi: 10.1109/ACCESS.2024.3351944.

[61] S. Fugkeaw, S. Rattagool, P. Jiangthiranan, and P. Pholwiset, "FPRESSO: Fast and Privacy-Preserving SSO Authentication With Dynamic Load Balancing for Multi-Cloud-Based Web Applications," *IEEE Access*, vol. 12, pp. 157888–157900, 2024, doi: 10.1109/ACCESS.2024.3485996.

[62] S. S. Mangalampalli *et al*., "Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (A3C) Algorithm in Multi Cloud Environment," *IEEE Access*, vol. 12, pp. 11354–11377, 2024, doi: 10.1109/ACCESS.2024.3355092.

[63] B. M. Beena *et al*., "Adaptive Energy Optimization in Cloud Computing Through Containerization," *IEEE Access*, vol. 13, pp. 159548–159565, 2025, doi: 10.1109/ACCESS.2025.3606932.

[64] V. C. Bharathi, S. Syed Abuthahir, M. Ayyavaraiah, G. Arunkumar, U. Abdurrahman, and S. Asad Ali Biabani, "O2O-PLB: A One-to-One-Based Optimizer With Priority and Load Balancing Mechanism for Resource Allocation in Fog-Cloud Environments," *IEEE Access*, vol. 13, pp. 22146–22155, 2025, doi: 10.1109/ACCESS.2025.3536210.

[65] J. Jeon, S. Park, B. Jeong, and Y.-S. Jeong, "Efficient Container Scheduling With Hybrid Deep Learning Model for Improved Service Reliability in Cloud Computing," *IEEE Access*, vol. 12, pp. 65166–65177, 2024, doi: 10.1109/ACCESS.2024.3396652.

[66] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *IEEE Access*, vol. 11, pp. 31866–31879, 2023, doi: 10.1109/ACCESS.2023.3262138.

[67] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-aho, "Deep Reinforcement Learning for Orchestrating Cost-Aware Reconfigurations of vRANs," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 200–216, Feb. 2024, doi: 10.1109/TNSM.2023.3292713.

[68] G. Volpe, A. M. Mangini, and M. P. Fanti, "A Deep Reinforcement Learning Approach for Competitive Task Assignment in Enterprise Blockchain," *IEEE Access*, vol. 11, pp. 48236–48247, 2023, doi: 10.1109/ACCESS.2023.3276859.

[69] S. Aleyadeh, A. Moubayed, P. Heidari, and A. Shami, "Optimal Container Migration/Re-Instantiation in Hybrid Computing Environments," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 15–30, 2022, doi: 10.1109/OJCOMS.2022.3140272.

[70] H. T. Nguyen, T. Van Do, and C. Rotter, "Scaling UPF Instances in 5G/6G Core With Deep Reinforcement Learning," *IEEE Access*, vol. 9, pp. 165892–165906, 2021, doi: 10.1109/ACCESS.2021.3135315.

[71] Reiss, C., Tumanov, A., Ganger, G.R., Katz, R.H. and Kozuch, M.A., 2012. *Heterogeneity and dynamicity of clouds at scale: Google cluster trace analysis*. Proceedings of the 3rd ACM Symposium on Cloud Computing (SoCC '12), pp. 1–13.

[72] Gan, Y., Zhang, Y., Hu, L., Cheng, P., He, Y., Zhou, J., Zheng, X., Hu, S., Katz, R.H. and Stoica, I., 2019. *An open-source benchmark suite for microservices and their hardware-software implications*. Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19), pp. 3–18.