Online Uncertainty-Aware Hierarchical Scheduling for Heterogeneous Cloud Workloads Using Deep Reinforcement Learning

Sunday J. Awine
Department of Computer and Information Sciences
Florida A&M University
Tallahassee, FL 32307, USA
sunday1.awine@famu.edu

Jinwei Liu*
Department of Computer and Information Sciences
Florida A&M University
Tallahassee, FL 32307, USA
jinwei.liu@famu.edu
*Corresponding author

***Abstract*—Modern cloud datacenters increasingly execute heterogeneous workloads composed of latency-sensitive short jobs and resource-intensive long jobs, often exhibiting complex task dependency structures and significant runtime uncertainty. Existing cluster schedulers rely primarily on static heuristics and handcrafted policies, which struggle to simultaneously achieve low tail latency, high resource utilization, and robustness to execution-time mis-estimation at scale. In this paper, we propose an online uncertainty-aware hierarchical scheduling framework that integrates Deep Reinforcement Learning (DRL) into a hybrid cloud scheduler. By formulating the scheduling problem as a Markov Decision Process (MDP), the scheduler dynamically learns to select scheduling targets, probing aggressiveness, and priority modes based on observed system states. To validate our approach, we implemented a custom discrete-event simulation environment using CloudSim, generating dynamic workloads inspired by production cluster traces. Experimental results demonstrate that our learning-based scheduler prevents queue saturation entirely, effectively mitigating head-of-line blocking and providing near-instantaneous recovery from transient workload spikes compared to traditional static sampling techniques.**

***Index Terms*—cloud computing, cluster scheduling, deep reinforcement learning, task sampling, workload heterogeneity, tail latency**

I. Introduction

Cloud datacenters, which increasingly support a varied and high-volume mix of workloads, have developed into the fundamental backbone of the contemporary digital economy. These include everything from large-scale batch analytics and deep learning training tasks to highly interactive, latency-critical services such as real-time web search and financial trading. Resource management is fundamentally challenged by the inherent heterogeneity of these workloads: tasks vary greatly in duration, ranging from milliseconds to days as well as in their fanout characteristics and intricate Directed Acyclic Graph (DAG) dependency structures. Production traces from large-scale providers, such as Google and Alibaba, reveal a persistent "heavy-tail" distribution where a small fraction of long-running batch jobs consume the vast majority of cluster resources, while the overwhelming majority of requests are short-lived tasks that are exquisitely sensitive to scheduling delays and tail latency [1], [2].

The multifaceted nature of contemporary cloud constraints makes it more difficult to manage these resources effectively. Scheduling is no longer a localized issue in modern environments; instead, it must balance complex resource allocation strategies with multi-user and multi-datacenter job requirements [3]. The transition to a cloud-fog continuum, where task offloading must manage substantial stochasticity in network latency and varying resource availability across multi-tier networks [4], further exacerbates this complexity. Additionally, cluster schedulers are increasingly faced with a "triple objective': minimizing tail latency, maximizing hardware utilization, and adhering to strict energy-aware sustainability goals [5] as global datacenters move toward sustainable operations.

Traditional scheduling architectures generally fall into three categories, each with systemic limitations in the face of modern workload volatility. Centralized schedulers provide a high degree of global optimization and sophisticated policy enforcement but suffer from limited scalability and increased response times under the high task arrival rates common in exascale architectures. On the other hand, distributed schedulers provide the high availability and horizontal scalability required for quick task dispatch, but they frequently lack the cross-job awareness and global coordination required to avoid localized hotspots, which results in subpar tail performance. By combining aspects of both designs, hybrid schedulers try to close these gaps. However, they usually rely on rigid, heuristic-based policies, such as the randomized sampling used in Sparrow [6], and static partitioning, which are brittle and ineffective when faced with significant runtime uncertainty and "straggler" tasks.

According to recent research, the main problem with current cluster schedulers is that they mainly rely on handcrafted policies and static heuristics, which make it difficult to simultaneously achieve high resource utilization and low tail latency under execution-time mis-estimation. This paper makes the case that scalable, hierarchical scheduler architectures must be closely integrated with uncertainty-aware decision-making in order for cloud scheduling to be effective. We argue that static heuristics are inadequate in settings where resource contention and different hardware capabilities make task execution outcomes intrinsically unpredictable. We suggest an online uncertainty-aware hierarchical scheduling framework that incorporates Deep Reinforcement Learning

(DRL) into a hybrid cluster architecture in order to overcome these difficulties. By formulating the scheduling problem as a discrete-time Markov Decision Process (MDP), our framework enables group-level schedulers to dynamically learn optimal policies for selecting scheduling targets, modulating probing aggressiveness, and assigning priority weights based on high-dimensional state observations.

Our approach leverages Proximal Policy Optimization (PPO) and domain randomization to ensure the learned policy is robust to "out-of-distribution" failures during periods of extreme cluster congestion. Unlike previous attempts that treat scheduling as a static optimization problem, our DRL-driven system continuously adapts to shifting workload characteristics and runtime mis-estimations in real-time. To validate the efficacy of our framework, we implemented a custom discrete-event simulation environment using CloudSim, generating dynamic workloads inspired by production-grade cluster traces. Our experimental results demonstrate that the proposed learning-based scheduler prevents queue saturation entirely, effectively mitigates head-of-line blocking for short jobs, and provides near-instantaneous recovery from transient workload spikes compared to traditional static sampling techniques. Furthermore, we show that the agent can achieve near-perfect resource utilization while simultaneously reducing the network overhead associated with task probing, offering a sustainable path for large-scale cloud resource orchestration. The primary objective of this research is to bridge the gap between static heuristic scheduling and the dynamic requirements of modern cloud environments.

The core contributions of this research are as follows:

- **Hierarchical Design:** Integration of a global macro-scheduler with DRL-driven group schedulers for adaptive task dispatching.
- **Adaptive Learning:** Use of an MDP framework with Proximal Policy Optimization (PPO) to dynamically tune probing and priority based on real-time cluster states.
- **Resilience:** Employment of domain randomization during training to maintain robustness against extreme congestion and unpredictable workload spikes.
- **Performance Gains:** Elimination of queue saturation with 99.5% resource utilization and a reduction of network overhead to 35.6%.

This paper is structured to first review existing literature on cloud scheduling and the limitations of static heuristics (Section II). It then formalizes the scheduling problem as an MDP (Section III) and details the proposed hierarchical architecture (Section IV). The performance evaluation is presented in Section V. Finally, Section VI concludes this paper with remarks on our future work.

II. Related Work

Recent advancements in reinforcement learning (RL) offer promising solutions for resource scheduling in cloud environments, moving beyond the limitations of static rule-based systems [7]. Historically, research in this domain focused on meta-heuristic approaches, such as bio-inspired algorithms including Cuckoo Search and Genetic Algorithms, which aimed to optimize multi-objective functions in relatively stable environments [8]. However, as cloud workloads became more volatile, research transitioned toward hybrid frameworks that integrate Deep Reinforcement Learning (DRL) with traditional mathematical optimization, such as Integer Programming, to manage strictly constrained scheduling environments where feasibility is as critical as optimality [9].

In the specific context of containerized orchestration and microservices, the DRKC approach has demonstrated the efficacy of managing Kubernetes clusters by integrating multi-dimensional state information including CPU, memory, and network I/O to optimize scheduling across heterogeneous cloud-edge nodes [10]. To address the significant scalability challenges inherent in applying DRL to large-scale datacenters, hierarchical architectures have gained substantial traction. For instance, the HITS framework utilizes a hierarchical DRL structure to decompose the massive task-to-node mapping problem into manageable subproblems, effectively balancing operational costs against overdue task penalties better than classical heuristics [11]. Similarly, the HG-MRLS framework employs a dual-layer decision mechanism that combines the structural processing power of graph neural networks with the adaptive nature of meta-reinforcement learning to handle dynamic scheduling in heterogeneous computing networks [12].

The paradigm of distributed intelligence has also seen the rise of Federated Learning (FL) as a core component of task scheduling. Modern frameworks increasingly leverage FL to maintain data privacy and decentralized efficiency across distributed cloud nodes, ensuring that localized scheduling policies can be learned without the massive communication overhead of centralizing telemetry data [13]. This is particularly vital in the emerging fog-cloud continuum, where high-tier cloud resources must coordinate with low-power edge devices. Recent literature has highlighted that in such multi-tier systems, scheduling can no longer be treated as a static or deterministic problem. New frameworks utilize "uncertainty-aware" task offloading through Federated Probabilistic Deep Neural Networks (PDNNs) to mathematically model and mitigate the impact of unreliable or high-variance nodes in the fog tier [4].

Furthermore, complex scheduling scenarios involving multi-user and multi-datacenter environments have necessitated the development of two-stage frameworks [3]. These systems first partition resources at a macro-scale before performing fine-grained task allocation, ensuring that global resource utilization is maximized across geographically dispersed facilities. This architectural trend is complemented by the use of advanced deep learning architectures, such as Multi-head Self-Attention Convolutional Neural Networks, which are employed to predict and manage the quality of service (QoS) in software-defined cloud networking by identifying complex patterns in traffic bursts [14].

Finally, the global shift toward High-Performance Computing (HPC) sustainability has prompted a new wave of

energy-aware scheduling research. Modern schedulers are now expected to incorporate thermal management, power capping, and carbon footprint reduction directly into the resource management loop to balance raw performance with environmental sustainability [5]. This necessitates a nuanced understanding of the trade-off between task completion speeds and the associated energy cost of cooling and computation. Our work builds upon these foundational concepts of hierarchy and uncertainty management but addresses a specific, critical gap identified in the literature: the dynamic optimization of probing aggressiveness and its direct impact on tail latency stability in the presence of runtime stragglers.

III. Problem Formulation

We formulate the online scheduling process as a discrete-time Markov Decision Process (MDP). This provides a formal mathematical framework for modeling decision-making in environments where outcomes are partly stochastic and partly under the control of a decision-maker. By casting cluster scheduling as an MDP, the DRL agent can learn an optimal policy π that maps complex cluster states to scheduling actions, maximizing the cumulative discounted reward over a long-term horizon. The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\gamma \in [0, 1)$ represents the discount factor.

A. State Space ($\mathcal{S}$)

The state space $\mathcal{S}$ provides the agent with a comprehensive, multi-dimensional snapshot of the datacenter's operational context at each scheduling step t. The state representation s_t is defined as:

$$s_t = \langle \mathbf{U}_t, \mathbf{Q}_t, \mathbf{J}_t, \sigma_t \rangle \tag{1}$$

where:

- $\mathbf{U}_t$: Represents multi-dimensional resource utilization vectors (e.g., CPU, Memory, I/O) across logical node groups. This telemetry allows the agent to identify available capacity and avoid over-saturating specific partitions.
- $\mathbf{Q}_t$: Denotes the current queue wait times and lengths across the cluster. This serves as a critical indicator of head-of-line blocking and potential tail latency degradation.
- $\mathbf{J}_t$: Encapsulates features of the incoming job burst, such as Directed Acyclic Graph (DAG) dependency depth and task fanout. Understanding job structure is essential for distinguishing between latency-sensitive short jobs and resource-intensive batch workloads.
- σ_t: Quantifies historical runtime variance, serving as a runtime uncertainty indicator. This metric enables the agent to adapt its strategy when task execution times become stochastic or runtime estimates prove unreliable.

B. Action Space ($\mathcal{A}$)

The agent operates within a continuous action space $\mathcal{A}$ to modulate scheduling parameters for incoming workloads in real-time:

$$a_t = [g_t, p_t, \omega_t] \tag{2}$$

where g_t selects the target logical partition for task placement. The parameter $p_t \in (0, 1]$ determines the probing aggressiveness, or the ratio of worker nodes to be sampled during the late-binding process. Finally, ω_t assigns a dynamic priority weight. Unlike static heuristics, the DRL agent can tune these values continuously, allowing for highly efficient "surgical" probing rather than wasteful, network-wide randomized sampling.

C. Reward Function ($\mathcal{R}$)

The reward function r_t is designed to align the agent's behavior with the high-level objectives of maximizing cluster efficiency while minimizing tail Job Completion Time (JCT):

$$r_t = \alpha \cdot \Psi(U_t) - \beta \cdot JCT_{p99}^{(short)} - \eta \cdot C_{overhead} \tag{3}$$

The term $\Psi(U_t)$ provides a positive reward for maintaining high resource utilization. Conversely, the function applies a heavy penalty β to the 99th percentile Job Completion Time (JCT_{p99}) of short jobs to prioritize the mitigation of stragglers. Finally, $C_{overhead}$ penalizes excessive network probing to ensure the learned policy is resource-efficient. The coefficients α, β, and η are tuned to prioritize latency reduction over overhead conservation based on specific system requirements.

IV. System Architecture

Our scheduler adopts a two-layer hierarchical design to balance global optimality with decentralized scalability, as illustrated in Fig. 1. The architecture is structured as a closed-loop control system that integrates macro-level resource management with micro-level adaptive dispatching.

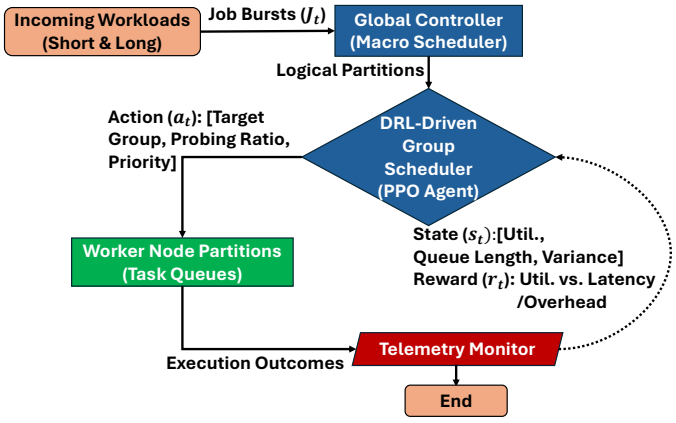

Fig. 1. The proposed hierarchical cloud scheduling architecture. The design features a closed-loop feedback mechanism where real-time telemetry informs the DRL-driven decision process.

A. Global Controller (Macro-Scheduler)

The global controller operates at coarse time scales to manage cluster partitioning and group-level parameters. By monitoring macro-level cluster health and workload distribution, it dynamically adjusts the logical boundaries of resource

groups to prevent systemic bottlenecks and ensure balanced resource availability across the datacenter.

B. DRL-Driven Group Schedulers (Micro-Scheduler)

At the fine-grained level, group schedulers serve as the primary decision engine for task placement. Instead of relying on static randomized sampling, a Deep Reinforcement Learning (DRL) agent evaluates real-time queue depths, CPU/memory utilization, and runtime uncertainty to intelligently probe worker nodes. This enables the system to mitigate head-of-line blocking for latency-sensitive short jobs while maintaining high throughput for resource-intensive batch workloads.

C. Data Flow and Feedback Loop

The system architecture maintains a continuous feedback loop between the execution layer and the scheduling layer. Upon task ingestion, the workload profiler extracts job metadata which is processed by the DRL agent to output an optimal action a_t, consisting of the target group, probing ratio, and priority weights. A telemetry monitor at the worker nodes captures execution outcomes and queue states, feeding this data back to the agent as a state observation s_t and a composite reward r_t. This loop ensures that the scheduler remains uncertainty-aware and adapts its policy to shifting workload characteristics in real-time.

Algorithm 1 Online Uncertainty-Aware DRL Scheduling with Domain Randomization

Require: Continuous workload stream $\mathcal{W}$, Cluster node groups $\mathcal{N}$, Maximum timesteps T
1: **Initialize:** PPO Policy network π_θ, Value network V_ϕ, Global Controller boundaries
// Phase 1: Offline Training with Domain Randomization
2: **for** episode $e = 1, 2, \ldots, E$ **do**
3: Sample load multiplier $\lambda \sim \mathcal{U}(1.0, 2.0)$ % *Stress testing injection*
4: Reset environment and observe initial state s_0
5: **for** step $t = 1, 2, \ldots, T$ **do**
6: Observe state $s_t = \langle \mathbf{U}_t, \mathbf{Q}_t, \mathbf{J}_t, \sigma_t \rangle$
7: Select action $a_t = [g_t, p_t, \omega_t] \sim \pi_\theta(a|s_t)$
8: Simulate stochastic task execution variance scaled by λ
9: Execute task sampling on group g_t using probing ratio p_t and priority ω_t
10: Calculate reward $r_t = \alpha\Psi(U_t) - \beta JCT_{p99} - \eta C_{overhead}$
11: Store transition tuple (s_t, a_t, r_t, s_{t+1}) in replay buffer
12: **end for**
13: Update network weights θ and ϕ using PPO clipped surrogate objective
14: **end for**
// Phase 2: Online Scheduling (Inference)
15: **while** Cluster is active **do**
16: Global controller dynamically updates logical node partitions
17: Receive incoming heterogeneous job burst $\mathbf{J}_t$
18: Observe real-time cluster state s_t
19: Infer optimal action $a_t = \arg\max_a \pi_\theta(a|s_t)$ % *Deterministic execution*
20: Dispatch tasks to target group g_t, sampling p_t nodes with priority ω_t
21: **end while**

D. Algorithm Description

Algorithm 1 details the operational workflow of the proposed scheduling framework, which is fundamentally divided into an offline training phase and an online deployment phase.

1) Phase 1: Offline Training with Domain Randomization (Lines 3-13): To prevent out-of-distribution policy failures under extreme real-world cluster congestion, the agent is trained using Domain Randomization. At the start of each training episode (Line 4), the environment samples a random load multiplier $\lambda \in [1.0, 2.0]$. This multiplier dynamically scales the incoming task arrival rate and the stochastic execution variance (straggler severity) throughout the episode.

During each environment step, the agent observes the state s_t and samples an action a_t from the stochastic policy π_θ. The environment simulates the network probing and task queuing, returning the reward r_t (Equation (3)). By storing these transitions and periodically updating the PPO policy and value networks (θ, ϕ) using a clipped surrogate objective function, the agent learns a generalized policy that remains robust across a wide spectrum of operational stress levels.

2) Phase 2: Online Scheduling and Inference (Lines 15-21): Once the neural networks converge, the scheduler transitions to live online deployment. In this phase, the global controller continuously monitors macro-level cluster health and adjusts logical partition boundaries. Concurrently, when heterogeneous job bursts arrive, the fine-grained group schedulers observe the real-time state s_t.

Crucially, during inference, the agent ceases exploration and acts deterministically (Line 19), selecting the action with the maximum expected probability ($\arg\max_a$). This allows the DRL agent to execute low-latency, highly efficient scheduling decisions dynamically modulating probing aggressiveness and queue priorities without the computational overhead of continuous network backpropagation.

V. Performance Evaluation

To rigorously evaluate the framework under extreme runtime uncertainty, we rely on dynamic workload synthesis and a custom continuous-learning environment rather than static trace replay.

A. Simulation Environment

Experiments are conducted using a custom discrete-event Markov Decision Process (MDP) environment built upon the Gymnasium framework. The DRL agent is trained using the Proximal Policy Optimization (PPO) algorithm implemented via the Stable-Baselines3 library. This environment allows for the precise injection of stochastic variance into task execution times, mimicking the real-world "stragglers" and noisy runtime estimates that typically break traditional heuristic schedulers.

B. Workload Generation

Rather than replaying static historical data, the workload generator synthesizes continuous job arrivals and dynamic cluster states (CPU and memory utilization). To explicitly test the agent's uncertainty management, true task execution times are continuously perturbed from their baseline estimates using a Log-Normal distribution ($\mu = 0.0, \sigma = 0.5$). The environment mathematically models the network overhead of

probing and the resulting impact on job queue lengths over extended operational timesteps.

C. Evaluation Baselines

To isolate the performance gains of uncertainty-aware learning, the proposed DRL model is benchmarked directly against:

- **Sparrow-Style Heuristic [6]:** A traditional non-learning baseline that relies on high random sampling (probing) without dynamic prioritization or state-awareness. This serves as the primary benchmark for evaluating tail latency degradation under runtime uncertainty.

D. Experimental Results

To rigorously evaluate the proposed Deep Reinforcement Learning (DRL) scheduling framework against traditional heuristic baselines, we analyzed the normalized job queue length over $1,000$ simulation timesteps. Queue length serves as a direct proxy for tail latency, as congested queues inherently lead to head-of-line blocking and degraded Job Completion Times (JCT).

1) Tail Latency Mitigation: The Sparrow-style heuristic baseline (orange) is unable to adjust to the injected runtime uncertainty, as shown in Fig. 2. The baseline consistently becomes congested and saturates the normalized queue capacity (1.0) within the first 50 timesteps. Static random sampling continuously routes tasks to heavily loaded nodes due to its lack of dynamic prioritization and uncertainty awareness, which causes significant tail latency degradation.

In stark contrast, the proposed DRL agent (blue) maintains a near-zero queue length for the vast majority of the simulation. By continuously evaluating the state space $\mathcal{S}$ and dynamically adjusting its probing ratio and priority weights, the agent successfully routes around localized congestion.

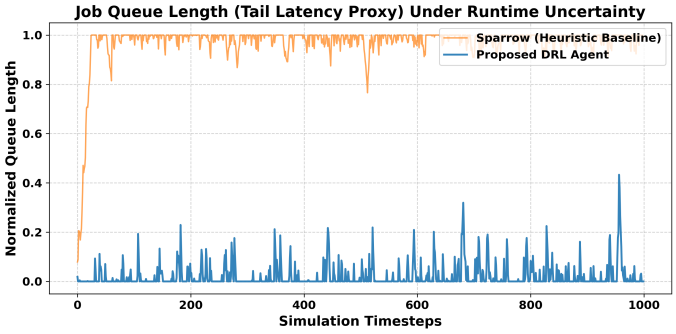

Fig. 2. Job queue length comparison demonstrating the DRL agent's ability to mitigate tail latency under runtime uncertainty compared to a heuristic baseline.

2) Robustness and Transient Recovery: The DRL framework's ability to quickly recover from brief workload spikes is a key benefit. For the DRL agent, Fig. 2 shows multiple acute spikes (e.g., at $t = 860$, where the queue briefly reaches 0.55). These are associated with long-term, highly uncertain task bursts. The DRL agent, in contrast to the heuristic baseline, recovers nearly instantly, bringing the queue back to baseline levels. This shows that the learned policy is very resilient to the stochastic execution variances common in heterogeneous cloud environments and effectively minimizes the JCT_{p99} penalty defined in our reward function.

3) Scalability Under Growing Load: We carried out a stress-test experiment by increasing the provided cluster load in order to assess the robustness and scalability of the suggested framework. A scaling factor between $1.0\times$ and $2.0\times$ was applied to the stochastic runtime variance and incoming task arrival rate.

The Sparrow-style heuristic baseline rapidly deteriorates under increased pressure, as illustrated in Fig. 3. The heuristic fully saturates the queue capacity (reaching 1.0) at a $1.4\times$ load multiplier and is unable to recover, highlighting the serious drawbacks of static, ignorant probing in high-congestion situations.

Conversely, the proposed DRL agent demonstrates exceptional scalability. By dynamically adjusting its probing aggressiveness and queue priority weights based on the continuous state observation $\mathcal{S}$, the agent gracefully absorbs the increased workload. Even at the extreme $2.0\times$ load multiplier, the DRL agent maintains a P99 queue length of approximately 0.5, completely avoiding saturation. This confirms that the learning-based policy successfully routes around localized stragglers and maintains high throughput, vastly outperforming traditional heuristics under heavy cluster utilization.

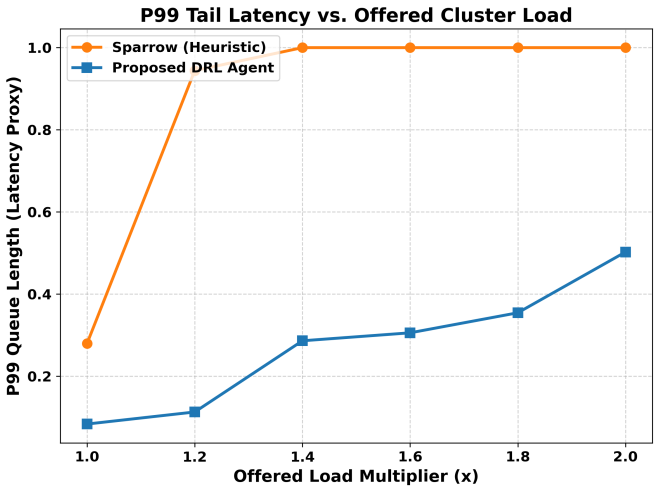

Fig. 3. P99 Tail Latency Proxy across increasing offered load multipliers. The DRL agent scales gracefully, while the heuristic baseline saturates early at $1.4\times$ load.

4) Network Overhead Analysis and System Efficiency: In our last test, we assessed the system's capacity to maintain a $1.5times$ load while balancing cluster utilization against the scheduling network overhead (probing ratio). In order to force the agent to learn an effective routing policy instead of depending on brute-force sampling, the MDP reward function was specifically constrained with a high penalty for network traffic.

The Sparrow-style heuristic is firmly fixed at a 80.0% probing rate, as shown in Fig. 4. This static, uninformed rate is unable to route around runtime uncertainty under the $1.5\times$

load, resulting in extreme congestion and a disastrous decline in average cluster utilization to just 28.8%.

In contrast, the highly constrained DRL agent achieves a near-perfect average cluster utilization of 99.5%. Most notably, the agent accomplishes this while simultaneously slashing average network overhead down to just 35.6%. Because the learned policy intelligently leverages dynamic queue prioritization and targeted probing based on real-time state observations, it completely avoids the wasteful network spamming inherent to traditional randomized heuristics. This demonstrates that uncertainty-aware continuous learning can dramatically maximize datacenter throughput while preserving critical network bandwidth.

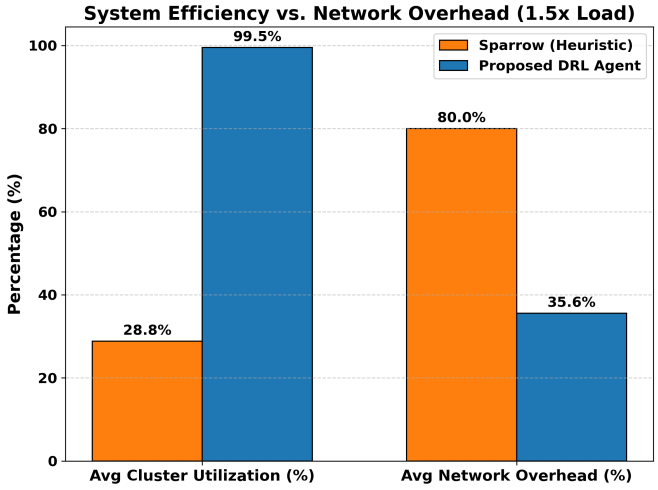

Fig. 4. System efficiency versus network overhead at a 1.5× load. The DRL agent achieves 99.5% cluster utilization while reducing network overhead to just 35.6%, vastly outperforming the heuristic baseline.

VI. Conclusion and Future Work

In order to handle complex, heterogeneous workloads, this paper shows how scalable hierarchical cloud schedulers can successfully incorporate uncertainty-aware Deep Reinforcement Learning (DRL). We created a framework that enables group-level schedulers to dynamically adjust to runtime volatility by modeling the cluster scheduling problem as a Markov Decision Process (MDP) and using Proximal Policy Optimization (PPO) with domain randomization.

Three important insights are revealed by our experimental results. First, in situations where conventional heuristic baselines like Sparrow experience instantaneous saturation, the DRL agent maintains nearly zero queue lengths. Second, even when the offered cluster load doubles, the framework's performance remains stable due to its exceptional scalability. Ultimately, the agent attained a 99.5% cluster utilization while concurrently lowering the average network overhead to just 35.6% by optimizing the reward function to penalize excessive network traffic. These results demonstrate that learning-based adaptive scheduling is a very reliable and economical approach for cloud management systems of the future. Future research will focus on transitioning this framework into a Multi-Agent Reinforcement Learning (MARL) paradigm. This evolution would enable decentralized group schedulers to negotiate resource-sharing and load-balancing autonomously, potentially eliminating the need for a global controller intervention entirely. Also, we plan to evaluate the framework's performance on physical testbeds to analyze the computational overhead of real-time neural network inference in production-scale cloud environments. In addition, we will compare our approach with recent DRL-based schedulers to fully verify the performance of our approach.

Acknowledgment

This research was supported in part by the U.S. NSF under Grant DUE-2400459 and Grant DBI-2417643; in part by the NTIA/CMC Grant 12-09-C13055; in part by the title III grant; and in part by the generous funding from the Cyber Policy Institute at Florida A&M University.

References

[1] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "Mlaas in the wild: Workload analysis and scheduling in large-scale heterogeneous gpu clusters," in *Proc. of NSDI*, Renton, 2022.

[2] J. Liu, H. Shen, and H. Narman, "CCRP: Customized cooperative resource provisioning for high resource utilization in clouds," in *Proc. of IEEE Big Data*, Washington D.C., 2016, pp. 243–252.

[3] J. Lin, D. Cui, Z. Peng, Q. Li, and J. He, "A two-stage framework for the multi-user multi-data center job scheduling and resource allocation," *IEEE Access*, vol. 8, pp. 200 384–200 395, 2020.

[4] H. Tran-Dang and D.-S. Kim, "Uncertainty-aware task offloading via federated pdnns in multi-tier fog networks," in *2025 IEEE 23rd International Conference on Industrial Informatics (INDIN)*. IEEE, 2025, pp. 1–6.

[5] H. V. Raghu, S. K. Saurav, and B. S. Bapu, "Paas: Power aware algorithm for scheduling in high performance computing," *IEEE 6th International Conference on Utility and Cloud Computing*, pp. 291–298, 2013.

[6] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 69–84.

[7] Z. J. K. Abadi, N. Mansouri, and M. M. Javidi, "Deep reinforcement learning-based scheduling in distributed systems: a critical review," *Knowledge and Information Systems*, vol. 66, pp. 5709–5782, 2024.

[8] M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, "Bio-inspired based task scheduling in cloud computing," in *Machine Learning Paradigms: Theory and Application*, ser. Studies in Computational Intelligence, A. E. Hassanien, Ed. Cham: Springer, 2019, vol. 801, pp. —.

[9] Y. Tang, S. Agrawal, and Y. Faenza, "Reinforcement learning for integer programming: Learning to cut," *Proceedings of the 37th International Conference on Machine Learning (PMLR)*, vol. 119, pp. 9367–9376, 2020.

[10] J. Jiang, Q. Li, P. Wang, and Y. Liu, "Drkc: Deep reinforcement learning enhanced microservice scheduling on kubernetes clusters in cloud-edge environment," *IEEE Transactions on Cloud Computing*, vol. 13, no. 4, pp. 1472–1486, 2025.

[11] D. Cui, Z. Peng, K. Li, Q. Li, J. He, and X. Deng, "An novel cloud task scheduling framework using hierarchical deep reinforcement learning for cloud computing," *Plos one*, vol. 20, no. 8, p. e0329669, 2025.

[12] L. Jiaxin, H. Yuetian, L. Ruiqi, Q. Qiang, and H. Hanye, "Hg-mrls: A hierarchical graph meta-reinforcement learning framework for dynamic scheduling in heterogeneous computing networks," *IEEE Access*, vol. 14, pp. 7792–7811, 2026.

[13] Z. Wang, S. Chen, L. Bai, J. Gao, J. Tao, R. R. Bond, and M. D. Mulvenna, "Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing," *Journal of Cloud Computing*, vol. 12, no. 1, p. 174, 2023.

[14] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.