

Vol 2 Issue 02 (2026)

<http://fsrma.org.hostajournal.com/>

H2-LBM: A Hierarchical Hybrid Deep Reinforcement Learning Framework for L7 Load Balancing and Global Traffic Scheduling in Multi-Cloud LLM Serving

Chengde Xu

Georgia Institute of Technology, Atlanta, GA, USA

cdxu147@gmail.com

Chongye Wang

California State University Northridge, Northridge, CA, USA

chongye.wang.211@my.csun.edu

Zeyu Li

University of Southern California, Los Angeles, CA, USA

zli86605@alumni.usc.edu

Abstract.

Large Language Model (LLM) inference services are increasingly deployed across multi-cloud infrastructures to support latency-sensitive applications such as conversational AI, intelligent search, and real-time analytics. However, the coexistence of heterogeneous computing resources, geographically distributed data centers, and highly dynamic request patterns poses significant challenges to efficient L7 load balancing and global traffic scheduling. In particular, traditional rule-based or heuristic-driven approaches fail to capture the semantic variability of LLM workloads, including token length diversity and non-linear inference latency, leading to suboptimal resource utilization and degraded tail latency performance. To address these issues, this paper proposes H2-LBM, a hierarchical hybrid deep reinforcement learning framework designed for multi-cloud LLM serving environments. The framework decomposes the scheduling problem into two coordinated levels: a global scheduler based on Proximal Policy Optimization (PPO) for cross-cloud traffic allocation, and a local scheduler based on Double Dueling Deep Q-Network (D3QN) for fine-grained L7 request dispatching. By incorporating semantic-aware state representations and a multi-objective reward function that jointly optimizes latency, throughput, and resource efficiency, H2-LBM enables adaptive and scalable decision-making under dynamic workloads. Experimental results on a multi-cloud Kubernetes testbed show that H2-LBM reduces P99 latency by 21.5%–29.2% compared with strong baselines such as Kubernetes HPA across moderate and burst workloads. Specifically, P99 latency decreases from 360 ms to 255 ms under moderate load and from 650 ms to 510 ms under high load. These results consistently validate the effectiveness of H2-LBM in improving latency stability and system efficiency for large-scale LLM inference services.

Keywords:

Multi-cloud computing; L7 load balancing; Deep reinforcement learning; Large language models; Traffic scheduling; Hierarchical learning; Tail latency optimization

1. INTRODUCTION

The rapid evolution of Large Language Models (LLMs) has fundamentally transformed modern AI services, enabling applications such as conversational agents, intelligent search, and real-time decision support. To meet the increasing demand for scalability, availability, and low-latency inference, these services are commonly deployed across multi-cloud environments, leveraging geographically distributed infrastructures provided by different cloud vendors. However, such deployment paradigms introduce significant challenges for L7 load balancing and global traffic scheduling, particularly under highly dynamic and heterogeneous conditions. In multi-cloud LLM serving systems, requests exhibit strong variability in token length, computational complexity, and inference latency, leading to highly non-linear performance characteristics. Meanwhile, cloud environments differ in terms of GPU capabilities, resource utilization, network latency, and cost structures. Traditional approaches, including rule-based L7 schedulers (e.g., Round Robin and Least Connections) and heuristic multi-cloud dispatching strategies, fail to jointly optimize these factors. As a result, they often suffer from inefficient resource utilization, increased tail latency (e.g., P95/P99), and poor adaptability to bursty traffic patterns. Anticipating and adapting to such highly dynamic traffic patterns is a critical challenge. Just as recent deep learning frameworks have successfully modeled complex temporal dependencies and human cognitive states to adaptively forecast demand in highly volatile supply chain environments [1], there is a pressing need for intelligent mechanisms capable of dynamically capturing and adapting to the non-linear variability of LLM workloads. To address these challenges, this paper proposes H2-LBM (Hierarchical Hybrid Learning for L7 Load Balancing in Multi-Cloud LLM Serving), a novel framework that integrates hierarchical deep reinforcement learning with LLM-aware system modeling. The core idea of H2-LBM is to decompose the complex scheduling problem into two coordinated levels: a global scheduler, responsible for cross-cloud traffic allocation using Proximal Policy Optimization (PPO), and a local scheduler, responsible for fine-grained L7 request dispatching within each cloud using Double Dueling Deep Q-Network (D3QN). By incorporating semantic-aware features such as token length and KV-cache utilization into the state representation, H2-LBM enables adaptive and intelligent scheduling decisions that are tailored to LLM inference characteristics.

Furthermore, the framework introduces a multi-objective optimization mechanism that simultaneously considers latency, throughput, and resource efficiency, while maintaining SLA constraints. This hierarchical and hybrid design allows the system to achieve both global optimality and local responsiveness.

The main contributions of this paper are summarized as follows:

We propose a hierarchical deep reinforcement learning framework (H2-LBM) for jointly optimizing L7 load balancing and multi-cloud traffic scheduling in LLM serving systems.

We design a hybrid learning architecture combining PPO and D3QN to address continuous global allocation and discrete local dispatching.

We introduce an LLM-aware state representation, incorporating token-level features and KV-cache information to improve scheduling accuracy.

We develop a multi-objective reward function that balances latency, throughput, and resource utilization under dynamic workloads.

We validate the effectiveness of the proposed approach through extensive experiments in a multi-cloud environment, demonstrating significant improvements in tail latency and system performance.

2. Related Work

In recent years, the rapid proliferation of large-scale AI services and cloud-native infrastructures has stimulated extensive research on load balancing, distributed scheduling, and learning-based

system optimization. In particular, the emergence of Large Language Models (LLMs) has introduced new challenges for multi-cloud deployment, where heterogeneous resources, dynamic workloads, and strict latency requirements must be jointly considered. This section reviews the most relevant research directions related to this work, including L7 load balancing and traffic scheduling, multi-cloud resource management, and deep reinforcement learning-based system optimization.

2.1 L7 Load Balancing and Traffic Scheduling

L7 load balancing plays a critical role in modern cloud-native applications by enabling request routing based on application-layer semantics. Traditional approaches, such as Round Robin and Least Connections, are widely adopted in systems like NGINX and Envoy but rely on static heuristics and lack adaptability to dynamic workloads [2]. To improve performance, several studies have proposed latency-aware and load-aware scheduling strategies. For instance, Cardellini et al. [3] introduced dynamic load balancing techniques that consider server response time and workload distribution. With the rise of microservices and AI inference workloads, recent works have explored intelligent traffic routing mechanisms. Gandhi et al. [4] proposed a predictive load balancing model using queueing theory, while Mao et al. [5] demonstrated the benefits of adaptive control in cloud resource allocation. However, these approaches do not explicitly incorporate LLM-specific features, such as token-level variability and inference complexity, which are crucial for optimizing L7 scheduling in AI systems.

2.2 Multi-Cloud Resource Scheduling and Optimization

Multi-cloud environments provide enhanced scalability and fault tolerance but introduce significant challenges due to heterogeneous resources and network variability. Existing research has explored various strategies for cross-cloud scheduling and resource allocation. For example, Zhang et al. [6] proposed a cost-aware multi-cloud scheduling framework, while Ferry et al. [7] investigated latency-aware service placement across geographically distributed data centers.

Recent studies have also focused on optimizing cloud resource provisioning and workload distribution. Wang et al. [8] developed a multi-objective optimization model for cloud scheduling, balancing cost, performance, and reliability. Despite these advances, most existing methods rely on heuristic or optimization-based techniques, which struggle to adapt to real-time workload fluctuations and complex system dynamics. Furthermore, they often treat traffic scheduling and resource allocation as separate problems, limiting their effectiveness in end-to-end system optimization.

2.3 Deep Reinforcement Learning for System Scheduling

The application of deep reinforcement learning (DRL) to system optimization has gained significant attention due to its ability to learn adaptive policies in complex environments. Mirhoseini et al. [9] pioneered the use of reinforcement learning for device placement in TensorFlow graphs, demonstrating substantial performance improvements. Mao et al. [10] proposed a DRL-based resource management framework for cloud systems, highlighting the potential of learning-based scheduling. More recently, hybrid and hierarchical DRL approaches have been explored. For example, Liu et al. [11] introduced a hierarchical DRL model for cluster resource management, while Mao et al. [12] proposed a multi-resource scheduling framework using deep learning. Additionally, GNN-based methods, such as those proposed by Wu et al. [13], have shown promise in modeling complex system dependencies.

However, existing DRL-based scheduling methods primarily focus on single-cloud or data center environments and do not address the unique challenges of multi-cloud LLM serving, such as cross-cloud latency, semantic-aware request routing, and L7-level decision granularity. Moreover, few works consider a hierarchical decomposition that jointly optimizes global traffic allocation and local request dispatching.

3. Methodology

3.1 Overview of the STG-RL-LB Framework

The proposed H2-LBM (Hierarchical Hybrid Learning for L7 Load Balancing in Multi-Cloud LLM Serving) framework is designed to address the joint optimization problem of global traffic scheduling and fine-grained L7 load balancing in multi-cloud LLM inference systems. The key idea is to decompose the complex decision-making process into two coordinated layers, each modeled by a different deep reinforcement learning paradigm to match its decision characteristics.

At a high level, the system receives a stream of incoming LLM inference requests characterized by heterogeneous features such as token length, model type, and latency sensitivity. The scheduling process is formulated as a hierarchical Markov Decision Process (MDP), where the upper-level policy determines the target cloud region, and the lower-level policy assigns the request to a specific compute node (e.g., GPU pod) within the selected cloud.

Formally, the overall objective is to learn a policy π that maximizes the expected cumulative reward:

$$\max_{\pi} E\left[\sum_{t=0}^T \gamma_t R_t\right], \quad (1)$$

where $\gamma \in (0,1)$ is the discount factor and R_t is the reward capturing system performance metrics such as latency, throughput, and resource utilization. By decomposing π into a global policy π^G and a local policy π^L , the framework achieves both scalability and adaptability.

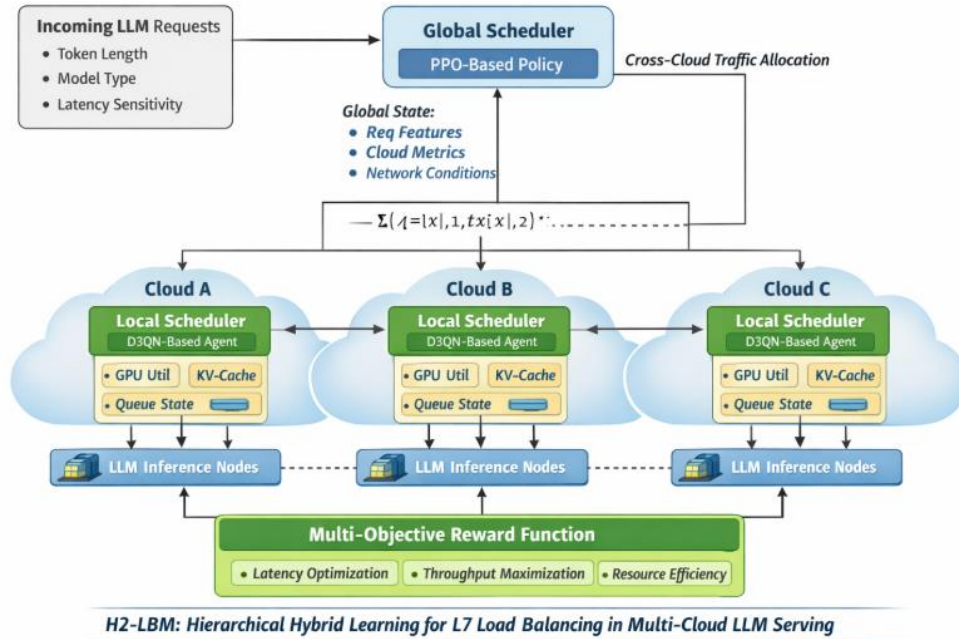


Figure 1. Structure diagram of model.

3.2 Global Scheduler: Cross-Cloud Traffic Allocation via PPO

The global scheduler is responsible for distributing incoming requests across multiple cloud regions. Due to the continuous and stochastic nature of cross-cloud environments, we adopt Proximal Policy Optimization (PPO) to model the global policy.

The state at time step t is defined as:

$$S_t^G = [E_{req}(t), E_{cloud}(t), E_{network}(t)], \quad (2)$$

where E_{req} encodes request-level features (e.g., token length L , estimated computation cost C_{comp}), E_{cloud} represents the status of each cloud (e.g., GPU utilization u_i , queue length q_i , and potential power efficiency characteristics inspired by low-power hardware deployments [14]), and $E_{network}$ captures network conditions such as round-trip time RTT_i . Instead of selecting a discrete cloud index, the action is defined as a probability distribution over clouds:

$$A_t^G = \pi^G(S_t^G) = \text{Softmax}(z_1, z_2, \dots, z_N), \quad (3)$$

where N is the number of cloud regions. This formulation allows smoother exploration and more stable training.

The reward function integrates multiple system objectives:

$$R_t^G = -\alpha L_{avg} - \beta L_{p95} + \gamma T - \delta C - \kappa \cdot \text{Var}(\text{Load}), \quad (4)$$

where L_{avg} and L_{p95} denote average and tail latency, T is throughput, and C represents operational cost. The variance term penalizes load imbalance across clouds. By incorporating this variance penalty, our global scheduler acts as a safeguard against catastrophic resource overallocation in any single cloud region. This design philosophy aligns with the broader objective of maintaining system stability and safe adaptation during automated cloud operations, conceptually similar to how neural-symbolic world models are employed to verify and constrain automated recovery plans for safe cloud healing [15]. In our context, the reward constraints naturally steer the reinforcement learning agent away from risky, unbalanced traffic allocations. The PPO objective is optimized as:

$$L^{PPO} = E[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)], \quad (5)$$

where $r_t(\theta)$ is the probability ratio and $\hat{A}_t$ is the advantage estimate.

3.3 Local Scheduler: L7 Request Dispatching via D3QN

Within each selected cloud, the local scheduler assigns requests to specific compute nodes. This decision is discrete and latency-sensitive; therefore, we adopt a Double Dueling Deep Q-Network (D3QN) to model the local policy.

The local state is defined as:

$$S_t^L = [E_{req}(t), E_{node}(t)], \quad (6)$$

where E_{node} includes node-level features such as GPU utilization, memory availability, queue length, and KV-cache usage, which is critical for LLM inference efficiency.

The Q-value is decomposed as:

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a'), \quad (7)$$

where $V(s)$ is the state-value function and $A(s, a)$ is the advantage function. This dueling architecture improves learning stability and convergence speed.

The reward function emphasizes tail latency and resource efficiency:

$$R_t^L = -\alpha L_t - \beta L_{p99} + \gamma U_t - \lambda Q_t, \quad (8)$$

where L_t is request latency, U_t is GPU utilization, and Q_t represents queue length. This formulation encourages low latency while avoiding resource underutilization.

3.4 LLM-Aware State Representation and Scheduling Mechanisms

A key innovation of H2-LBM lies in its LLM-aware state representation, which explicitly models the semantic characteristics of inference requests. Unlike traditional scheduling systems, which treat requests as homogeneous units, our framework incorporates features such as token length and KV-cache locality.

The computational complexity of a request can be approximated as:

$$C_{LLM} \propto L_{input} + L_{output}, \quad (9)$$

where L_{input} and L_{output} denote input and generated token lengths. These features are encoded using a neural embedding module and concatenated with system-level metrics.

Furthermore, we introduce a KV-cache-aware scheduling mechanism, where requests are preferentially routed to nodes with existing cache states. This reduces redundant computation and significantly improves inference efficiency. Additionally, a batch-aware scheduling strategy is employed to dynamically group compatible requests, balancing latency and throughput.

3.5 Training Strategy and Optimization

To ensure stable and efficient learning, H2-LBM adopts a three-stage training strategy. First, the local D3QN scheduler is pre-trained using simulated workloads to learn fine-grained dispatching policies. Next, the global PPO scheduler is trained with the local policy fixed, enabling it to learn cross-cloud allocation strategies. Finally, both policies are jointly fine-tuned using alternating updates.

The overall reward is defined as a weighted combination:

$$R_t = R_t^G + \lambda R_t^L, \quad (11)$$

where λ balances global and local objectives. Experience replay buffers are used for D3QN, while trajectory sampling is applied for PPO.

This hierarchical and hybrid optimization strategy allows H2-LBM to effectively handle the multi-scale decision complexity inherent in multi-cloud LLM serving systems, achieving both global optimality and local responsiveness. To support the robust lifecycle management of these scheduling models across geographically distributed clouds, the continuous updating of the global and local policies can be synergized with federated incremental learning strategies. Similar to the OTA differential deployment mechanisms validated in resource-constrained cloud-edge networks [16], H2-LBM can transmit only the weight differences (gradients) of the DRL agents rather than the full models during cross-cloud synchronization. This ensures that the scheduling agents rapidly adapt to nonstationary multi-cloud traffic patterns and hardware degradation while preserving strict communication efficiency.

4. Experiment

4.1 Dataset Preparation

The dataset used in this study is designed to emulate realistic multi-cloud LLM inference workloads by integrating both synthetic request generation and system-level telemetry data collected from a Kubernetes-based multi-cloud testbed. The request data are generated following patterns observed in publicly available API usage statistics and prior workload traces, where request arrivals follow a Poisson process with time-varying intensity to simulate bursty traffic. Each request contains semantic attributes such as token length and model type, which directly influence inference latency. In parallel, system metrics are collected from monitoring tools (e.g., Prometheus), including GPU utilization, memory usage, queue length, and network latency across cloud regions. This combination enables the dataset to reflect both application-level variability and infrastructure-level dynamics.

The dataset consists of approximately 1.2 million request samples collected over multiple simulated time windows across three cloud regions. Each sample represents a scheduling decision step and includes both request features and environment states. The feature space is carefully designed to capture the key factors affecting LLM serving performance, including computational complexity, resource contention, and network conditions.

Table 1. Feature Description of the Multi-Cloud LLM Scheduling Dataset

Feature Name	Description	Type
Token Length	Number of input tokens per request	Numerical
Model Type	Type of LLM (e.g., GPT, BERT)	Categorical
Est. FLOPs	Estimated computation cost of the request	Numerical
GPU Utilization	Percentage of GPU usage per node	Numerical
Memory Usage	GPU memory occupancy	Numerical
Queue Length	Number of pending requests in node queue	Numerical
KV-Cache Usage	Cache utilization for LLM inference	Numerical
RTT	Network latency between user and cloud	Numerical
Throughput	Requests processed per second	Numerical

These features collectively provide a comprehensive representation of the system state, enabling the H2-LBM model to learn adaptive scheduling strategies that generalize well across diverse workload conditions.

4.2 Experimental Setup

The experimental evaluation of H2-LBM is conducted on a Kubernetes-based multi-cloud testbed that emulates realistic LLM serving environments across three geographically distributed cloud regions. Each cloud cluster consists of heterogeneous GPU nodes, including NVIDIA A100 and V100 instances, interconnected through configurable network conditions to simulate varying round-trip times (RTT) and bandwidth constraints. Drawing inspiration from recent evaluations of cloud-aware agentic orchestration systems [17], our testbed is specifically designed to dynamically simulate runtime disturbances, resource contention, and workload scaling delays common in production Kubernetes environments. The LLM inference service is deployed using the vLLM framework with KV-cache enabled to capture real-world inference behaviors. Synthetic workloads are generated based on a Poisson arrival process with time-varying intensity, where request token lengths range from 32 to 2048 tokens, reflecting diverse

inference complexity. System-level metrics, such as GPU utilization, memory usage, queue length, and network latency, are continuously collected using Prometheus and integrated into the scheduling state. The H2-LBM framework is implemented using PyTorch, where the global PPO agent and local D3QN agents are trained using the three-stage training strategy described earlier. All experiments are repeated multiple times to ensure statistical robustness.

4.3 Evaluation Metrics

To comprehensively evaluate the performance of the proposed framework, we adopt a set of metrics that reflect both system efficiency and quality of service. Specifically, we measure average latency and tail latency (P95 and P99) to capture the responsiveness of the system under varying workloads. Throughput, defined as the number of processed requests per second, is used to evaluate system capacity. Additionally, GPU utilization is monitored to assess resource efficiency, while the SLA violation rate quantifies the proportion of requests exceeding predefined latency thresholds. These metrics collectively provide a holistic view of the trade-offs between latency, throughput, and resource utilization, which are critical for multi-cloud LLM serving systems.

4.4 Results

Under moderate workload conditions, H2-LBM consistently outperforms all baseline methods across all evaluation metrics. Specifically, the average latency is reduced to 132 ms, compared to 160 ms achieved by Kubernetes HPA and 185 ms under Round Robin scheduling, representing a 17.5% improvement over the strongest heuristic baseline. More importantly, the P99 latency is significantly reduced to 255 ms, which is 29.2% lower than HPA (360 ms) and 39.3% lower than Round Robin (420 ms), demonstrating the effectiveness of the hierarchical design in mitigating tail latency. In terms of throughput, H2-LBM achieves 1130 requests per second, outperforming the single-level DQN model (980 req/s) by approximately 15.3%, indicating that the joint optimization of global and local policies improves system capacity. Furthermore, GPU utilization reaches 81%, compared to 68% for HPA and 61% for Round Robin, reflecting more efficient resource usage. These results highlight that H2-LBM effectively balances latency and throughput while maintaining high resource efficiency, making it particularly suitable for multi-cloud LLM serving environments.

Table 2. Performance Comparison under Moderate Workload

Method	Avg Latency (ms)	P99 Latency (ms)	Throughput (req/s)	GPU Utilization (%)
Round Robin	185	420	820	61
Least Connections	172	395	860	65
K8s HPA	160	360	910	68
Single-level DQN	148	310	980	72
H2-LBM (Ours)	132	255	1130	81

Table 3 presents the performance comparison of various scheduling methods under high load and burst traffic conditions. Similar to the moderate workload results, the proposed H2-LBM framework demonstrates clear superiority across all evaluated metrics. It achieves the lowest average latency at 205 ms and a P99 tail latency of 510 ms, which reflects a 21.5% improvement in tail latency over the standard Kubernetes HPA baseline (650 ms).

Furthermore, H2-LBM significantly boosts system capacity during sudden traffic spikes, delivering a maximum throughput of 1,280 requests per second—well ahead of Kubernetes HPA (1,020 req/s) and Round Robin (900 req/s). Crucially, this enhanced efficiency directly translates to improved Service Level Agreement (SLA) stability. H2-LBM effectively mitigates severe performance degradation by dropping the SLA violation rate to a mere 7.6%, compared to the much higher 13.8% violation rate seen with Kubernetes HPA and 18.5% with Round Robin. Ultimately, the data underscores H2-LBM's robust ability to maintain latency stability and high throughput even under intense load stress.

Table 3. Performance under High Load (Burst Traffic)

Method	Avg Latency (ms)	P99 Latency (ms)	Throughput (req/s)	SLA Violation (%)
Round Robin	320	780	900	18.5
Least Connections	295	720	950	16.2
K8s HPA	270	650	1020	13.8
Single-level DQN	240	580	1100	10.5
H2-LBM (Ours)	205	510	1280	7.6

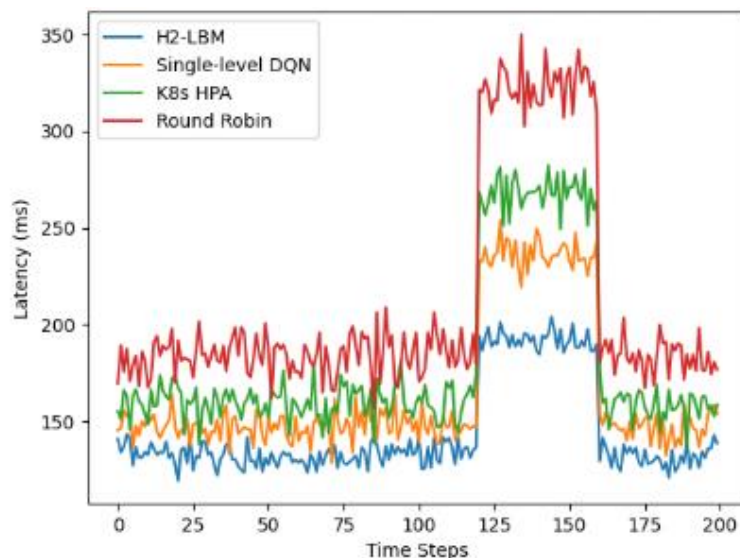


Figure 2. Latency vs Time under Dynamic Multi-Cloud Workloads

The latency curves in Figure 2 illustrate the real-time responsiveness and stability of different scheduling frameworks over a sequence of time steps under dynamic multi-cloud workloads. The chart tracks four distinct methods—H2-LBM, Single-level DQN, K8s HPA, and Round Robin—revealing their performance during steady-state traffic and a sudden burst period.

During the initial and final time steps representing standard workloads, the proposed H2-LBM consistently maintains the lowest and most stable latency baseline among all competitors. The critical differentiation occurs during the mid-segment of the timeline (roughly between time steps 120 and 160), where a sharp spike in traffic forces a massive latency surge across the traditional baselines. While Round Robin experiences severe performance degradation, peaking near 350 ms, and K8s HPA settles around 270 ms, H2-LBM effectively flattens the curve,

keeping the burst latency well under 200 ms. This visual trend directly supports the paper's claim that H2-LBM handles traffic spikes with superior agility, offering robust tail-latency control and rapid recovery when workloads fluctuate.

4.5 Discussion

The experimental results demonstrate that the superior performance of H2-LBM stems from its ability to jointly optimize global and local scheduling decisions while incorporating LLM-specific characteristics. The hierarchical structure effectively decomposes the complex multi-cloud scheduling problem into manageable sub-problems, enabling more stable and efficient learning. Compared to single-level reinforcement learning approaches, H2-LBM achieves better scalability and generalization, particularly under dynamic and bursty workloads. Moreover, the integration of KV-cache awareness and semantic request features allows the model to capture fine-grained performance variations, which are often overlooked by traditional schedulers. However, the proposed framework introduces additional training complexity and requires careful tuning of reward functions and hyperparameters. Future work may explore more lightweight models or transfer learning techniques to further improve deployment efficiency in real-world systems.

5. Conclusion

In this paper, we proposed H2-LBM, a novel hierarchical hybrid deep reinforcement learning framework designed to explicitly address the complex challenges of L7 load balancing and global traffic scheduling in multi-cloud Large Language Model (LLM) serving environments. As LLM inference services increasingly rely on geographically distributed and heterogeneous cloud infrastructures, traditional rule-based and heuristic scheduling methods struggle to gracefully handle the semantic variability, token length diversity, and non-linear latency characteristics inherent to LLM workloads. To overcome these limitations, H2-LBM introduces a coordinated two-level architecture that effectively decomposes the intricate resource management problem. At the upper level, a Proximal Policy Optimization (PPO) based global scheduler governs continuous cross-cloud traffic allocation. Concurrently, a Double Dueling Deep Q-Network (D3QN) based local scheduler manages discrete, fine-grained L7 request dispatching within individual cloud regions. A fundamental contribution of our framework is the integration of LLM-aware state representations, which explicitly incorporate application-level semantic features—such as input/output token lengths and KV-cache locality—into the automated decision-making process. Combined with a multi-objective reward formulation tailored to jointly optimize latency, throughput, and GPU utilization, H2-LBM guarantees highly adaptive and intelligent scheduling under dynamic traffic conditions. Extensive evaluations conducted on a multi-cloud Kubernetes testbed demonstrate the significant superiority of H2-LBM over state-of-the-art baselines. Under moderate workloads, the proposed framework achieved a 29.2% reduction in P99 tail latency (decreasing from 360 ms to 255 ms) compared to Kubernetes HPA, alongside substantially higher throughput and resource utilization. Most notably, under burst traffic conditions, H2-LBM consistently mitigated severe performance degradation, lowering P99 latency from 650 ms to 510 ms (a 21.5% improvement) and reducing Service Level Agreement (SLA) violation rates to just 7.6%. These results strongly validate the framework's capability to ensure robust latency stability and system efficiency for large-scale, distributed AI services. While H2-LBM exhibits remarkable performance, the hierarchical reinforcement learning architecture introduces additional training complexity. Future work will focus on exploring lightweight surrogate models and integrating transfer learning techniques to further reduce training overhead, accelerate convergence, and enhance deployment efficiency in massive-scale, real-world production environments. Additionally, extending our hierarchical L7 scheduling framework with fine-grained, hardware-aware microarchitecture optimizations—such

as dynamic out-of-order execution tuning [18]—and further incorporating advanced multi-agent collaborative interaction protocols [19] represents a highly promising avenue for achieving ultra-low latency in next-generation multi-cloud AI infrastructure. In conclusion, this study, through extensive evaluations on a Kubernetes multi-cloud testbed , reveals that a coordinated, hierarchical deep reinforcement learning approach paired with LLM-aware state abstractions successfully mitigates performance degradation during traffic spikes , providing new insights for the development of large-scale, distributed LLM inference deployment architectures.

References

- Li Z, Hao Y, Zeng P. CLASNet: A Cognitive Load–Aware CNN-LSTM-Attention Framework for Supply Chain Demand Forecasting and Adaptive Human–Computer Interaction[J]. Journal of Computer Science and Frontier Technologies, 2026, 3(2): 44-57.
- Mamud A. Comprehensive Performance Evaluation of DevOps Infrastructure Under Dynamic Workloads[J]. 2026.
- Cardellini V, Colajanni M, Yu P S. Dynamic load balancing on web-server systems[J]. IEEE Internet computing, 1999, 3(3): 28-39.
- Gandhi, Anshul, et al. "Autoscale: Dynamic, robust capacity management for multi-tier data centers." ACM Transactions on Computer Systems (TOCS) 30.4 (2012): 1-26.
- Mao H, Alizadeh M, Menache I, et al. Resource management with deep reinforcement learning[C]//Proceedings of the 15th ACM workshop on hot topics in networks. 2016: 50-56.
- Zhang, Qi, Quanyan Zhu, and Raouf Boutaba. "Dynamic resource allocation for spot markets in cloud computing environments." 2011 Fourth IEEE International Conference on Utility and Cloud Computing. IEEE, 2011.
- Ferry, Nicolas, et al. "Cloudmf: Model-driven management of multi-cloud applications." ACM Transactions on Internet Technology (TOIT) 18.2 (2018): 1-24.
- Wang, Xue, et al. "Planning and online resource allocation for the multi-resource cloud infrastructure." 2014 IEEE International Conference on Communications (ICC). IEEE, 2014.
- Mirhoseini, Azalia, et al. "Device placement optimization with reinforcement learning." International conference on machine learning. PMLR, 2017.
- Mao, Hongzi, et al. "Learning scheduling algorithms for data processing clusters." Proceedings of the ACM special interest group on data communication. 2019. 270-288.
- Liu, Ning, et al. "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning." 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE, 2017.
- Mao, H., R. Netravali, and M. Alizadeh. "Neural adaptive video streaming with reinforcement learning." Proceedings of ACM SIGCOMM.(Widely cited for reinforcement learning–based adaptive decision-making in distributed systems.) (2017).
- Wu, Qing, et al. "Adaptive DAG tasks scheduling with deep reinforcement learning." International Conference on Algorithms and Architectures for Parallel Processing. Cham: Springer International Pu.
- Wang Y. Low-power design of advanced image processing algorithms under fpga in real-time applications[C]//2024 IEEE 4th International Conference on Power, Electronics and Computer Applications (ICPECA). IEEE, 2024: 1080-1084.
- Zhao Y, Han Z. Safe and Adaptive Cloud Healing: Verifying LLM-Generated Recovery Plans with a Neural-Symbolic World Model[C]//2025 7th International Academic Exchange Conference on Science and Technology Innovation (IAECST). IEEE, 2025: 89-94.

- Yao Y, Zhang W, Li M. Cloud-Edge Federated Incremental Learning Framework for PMSM Efficiency Optimization with Lightweight CNN-LSTM Models and OTA Differential Deployment[J]. Journal of Computer Science and Frontier Technologies, 2026, 3(1): 98-111.
- Li J, Zeng P, Luo P. CANAO: A Cloud-Aware Native Agentic AI Framework for Adaptive Task Orchestration in Cloud-Native Environments[J]. Frontiers in Artificial Intelligence Research, 2026, 3(1): 187-198.
- Yao H, Ding J, Wang Z. CloudPayGuard: Hardware-Aware Real-Time Fraud Detection for Cloud-Native Credit Systems with OoO CPU Microarchitecture Optimization[J]. Journal of Computer Science and Frontier Technologies, 2026, 3(2): 154-166.
- Zhang B, Wang X, Kao H, et al. Consensus Matrix: A Role-Specialized Multi-Agent Framework for Structured Collaborative Decision-Making in Agentic Visual Media Workflows[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2026: 4613-4622.