SLA-MORL: SLA-Aware Multi-Objective Reinforcement Learning for HPC Resource Optimization

Seraj Al Mahmud Mostafa[1], Aravind Mohan[2], Jianwu Wang[1]
[1]*Department of Information Systems, University of Maryland, Baltimore County,* Baltimore, MD, USA
[2]*Department of Computer Science, McMurry University,* Abilene, TX, USA

***Abstract*—Dynamic resource allocation for machine learning workloads in cloud environments remains challenging due to competing objectives of minimizing training time and operational costs while meeting Service Level Agreement (SLA) constraints. Traditional approaches employ static resource allocation or single-objective optimization, leading to either SLA violations or resource waste. We present SLA-MORL, an adaptive multi-objective reinforcement learning framework that intelligently allocates GPU and CPU resources based on user-defined preferences (time, cost, or balanced) while ensuring SLA compliance. Our approach introduces two key innovations: (1) intelligent initialization through historical learning or efficient baseline runs that eliminates cold-start problems, reducing initial exploration overhead by 60%, and (2) dynamic weight adaptation that automatically adjusts optimization priorities based on real-time SLA violation severity, creating a self-correcting system. SLA-MORL constructs a 21-dimensional state representation capturing resource utilization, training progress, and SLA compliance, enabling an actor-critic network to make informed allocation decisions across 9 possible actions. Extensive evaluation on 13 diverse ML workloads using production HPC infrastructure demonstrates that SLA-MORL achieves 67.2% reduction in training time for deadline-critical jobs, 68.8% reduction in costs for budget-constrained workloads, and 73.4% improvement in overall SLA compliance compared to static baselines. By addressing both cold-start inefficiency and dynamic adaptation challenges, SLA-MORL provides a practical solution for cloud resource management that balances performance, cost, and reliability in modern ML training environments. Our code is open-source and available at https://github.com/big-data-lab-umbc/SLA-MORL.**

***Index Terms*—SLA, MORL, Service-Level Agreement, Multi Objective Reinforcement Learning, Optimization**

I. Introduction

Modern High-Performance Computing (HPC) and Cloud environments increasingly support complex AI models that process datasets varying widely in size and complexity, necessitating careful alignment between available resources such as CPUs and GPUs, and workload requirements. Traditional management strategies struggle with dynamically matching resource allocation to these diverse and fluctuating demands, often leading to inefficient utilization, increased operational costs, and compromised performance, while over-provisioning expensive hardware drives operational costs skyward with marginal performance gains. Multi-Objective Reinforcement Learning (MORL), a RL mechanism [1] to tackle various objectives simultaneously, emerges as a promising solution by learning to adaptively optimize resources through real-time interactions while balancing multiple competing Service Level Agreement (SLA) objectives such as cost efficiency, training time, and resource utilization.

However, applying MORL to complex HPC and Cloud environments introduces significant challenges. First, RL algorithms suffer from sample inefficiency [2], requiring extensive interactions with dynamic systems, thus incurring high computational costs and prolonged training periods [3]. Second, traditional RL methods fail to rapidly adapt to workload pattern changes and demand spikes, leading to costly overprovisioning or detrimental SLA violations [4]. Third, multi-objective optimization compounds these issues as RL policies typically prioritize single metrics, neglecting crucial Quality of Service (QoS) tradeoffs and resulting in resource hoarding [5]. Furthermore, RL agents lack built-in monitoring for resource misuse while inadequate state representations and limited resource capability knowledge lead to suboptimal configurations [3]. Finally, algorithmic instability, poor sim-to-real generalization, and exploration-exploitation imbalances further hinder deployment, risking frequent retraining and eroded user trust [6].

Current resource optimization approaches suffer from either static allocation strategies that ignore dynamic workload patterns, or reactive adaptive methods that lack workload-awareness and SLA coordination, resulting in inefficient resource utilization, increased operational costs, and compromised performance. We identify two critical research challenges: **(1)** *Cold-Start and Exploration Inefficiency in Existing Methods*: Traditional approaches begin with random resource allocations and lack mechanisms to leverage historical patterns, forcing costly exploration from scratch for every new workload; **(2)** *Static Multi-Objective Optimization*: Current systems employ fixed weight parameters for competing objectives, failing to adapt dynamically to SLA violations and changing system states.

To tackle these challenges, we introduce **SLA-MORL**, a multi-objective reinforcement learning framework that accepts model-data pairs and user-defined SLA preferences as input, and produces optimized resource allocation solutions with comprehensive performance analytics. The framework addresses dynamic resource allocation through two key innovations: **(1)** *Intelligent Initialization with Optional Historical*

Learning: SLA-MORL eliminates cold-start problems through an adaptive actor-critic architecture that either pre-trains from historical logs when available or employs efficient baseline runs (10% epochs, 20% data) for new workloads, ensuring intelligent starting configurations without costly random exploration; **(2)** *SLA-aware Pareto Optimization*: Our novel contribution unifies SLA violation detection and adaptive weight computation into a dynamic reward system that automatically adjusts optimization priorities based on violation severity.

SLA-MORL supports three user preferences: *time* (minimize training time using maximum resources where cost can increase), *cost* (minimize expense while allowing extended training time), and *balanced* (optimize both objectives with optional user-specified targets). This adaptive capability is formalized through our core optimization objective:

$$a_t = \arg\max_{a \in \mathcal{A}} \sum_{i \in \{time, cost, util\}} w_i^{(t)} R_i(s_t, a), \qquad (1)$$

where $w_i^{(t)}$ represent adaptive weights that evolve according to SLA violation patterns and user preferences, s_t captures workload characteristics and system state enhanced with preference encoding, and $a \in \mathcal{A}$ represents resource configurations (GPU, CPU).

We evaluate SLA-MORL on 13 diverse ML workloads using production HPC infrastructure with NVIDIA RTX 8000 GPUs. Our experiments demonstrate significant improvements over static baselines: 67.2% reduction in training time, 68.8% reduction in operational costs, and 73.4% improvement in SLA compliance. These results validate that our two technical innovations (historical learning for intelligent initialization and adaptive weight computation for SLA aware optimization) successfully address the fundamental challenges in cloud resource allocation for ML workloads.

II. Related Work

A. Resource Optimization with Reinforcement Learning

Recent advances in deep reinforcement learning have shown promise for dynamic resource allocation in cloud environments. Chen et al. [7] propose an A3C-based resource allocation method that combines actor-critic networks to optimize QoS and energy efficiency. While their approach demonstrates superior performance over traditional heuristics, it employs static weight parameters that cannot adapt to SLA violations. Similarly, Zhao et al. [8] present a DDPG-based task scheduling framework with correlation-aware state representations. However, both approaches optimize for fixed objectives without considering dynamic SLA requirements.

Liu et al. [9] advance the field with a hierarchical DRL framework combining global VM placement with LSTM-powered local server control, achieving significant energy and latency reductions. Baheri et al. [10] introduce MARS, an SLA-aware scheduler that combines offline heuristics with runtime A3C optimization. While MARS considers SLA constraints, it lacks our adaptive weight mechanism and suffers from cold-start problems when historical data is unavailable.

B. Offline to Online Learning Approaches

The integration of offline and online learning has emerged as a solution for improving sample efficiency in RL systems. Zheng et al. [11] introduce Adaptive Policy Learning (APL), where agents pre-train on static datasets before adapting with online feedback. Wang et al. [12] propose FamO2O, which pre-trains multiple policies and selects them adaptively per state. Ball et al. [2] present RLPD, accelerating online learning through symmetric sampling and ensemble methods.

These offline-to-online approaches primarily target robotics domains (Niu et al. [13]) or multi-agent environments (Liu et al. [14]). None address the unique challenges of cloud resource optimization where workloads vary dramatically and SLA violations carry immediate financial consequences. Our work uniquely combines historical learning with dynamic SLA-aware adaptation for cloud environments.

C. Multi-Objective Reinforcement Learning

Multi-objective optimization in RL has received significant attention for balancing conflicting goals. Nguyen et al. [15] propose two actor-critic methods (MCSP and SCMP) that integrate scalarized rewards into policy optimization. Liu et al. [16] introduce PSL-MORL using hypernetworks to generate personalized policies across the Pareto front. Cai et al. [17] extend Pareto-optimal MORL with distributional approaches using utility-based reward shaping, Rakshit et al. [18] introduces informed Pareto Simulated Annealing (iPSA) to efficiently explore the trade-offs in deployment configurations.

While these MORL methods advance the theoretical foundations, they operate primarily in simulated environments with static weight parameters. Zhou et al. [19] make progress with gradient-adaptive Pareto optimization for constrained RL, but their approach lacks integration with real-world SLA constraints. Qin et al. [20] address deadline-constrained workflow scheduling through DCMORL, yet still rely on fixed Chebyshev scalarization without adaptive mechanisms.

D. SLA-Aware Resource Management

SLA compliance represents a critical requirement in production cloud systems. Souza et al. [21] propose ASAX, an RL-based HPC co-scheduler using decision tree experts to maintain QoS constraints, improving utilization by up to 51%. Haratian et al. [22] present AFRM, a fuzzy resource management framework that reduces SLA violations through dynamic policies. Dai et al. [23] enhance constraint satisfaction through augmented proximal policy optimization.

However, these SLA-aware approaches share common limitations: they lack learning from historical patterns, operate with discrete action spaces (Ran et al. [24]), or focus on single objectives (Peng et al. [25]). None provide the adaptive weight mechanisms necessary for handling dynamic SLA violations in multi-objective settings.

E. Research Gap and Our Contribution

Existing approaches fall into three categories, each with critical limitations: **Single-objective RL methods** (Chen et

al. [7], Baheri et al. [10]) use static weights and cannot adapt when SLA violations occur. **General MORL frameworks** (Nguyen et al. [15], Liu et al. [16]) lack real-world cloud integration and operate in simulated environments. **SLA-aware systems** (Souza et al. [21], Dai et al. [23]) miss opportunities for historical learning and adaptive optimization.

SLA-MORL uniquely addresses these gaps through: (1) intelligent initialization that leverages historical patterns or efficient baseline runs to eliminate cold-start problems, and (2) dynamic weight adaptation based on SLA violation severity, enabling self-correcting resource allocation. Unlike existing work that treats these challenges separately, we provide an integrated solution specifically designed for production cloud environments where both rapid adaptation and SLA compliance are critical.

III. Methodology

We propose SLA-MORL, a dynamic resource allocation framework for ML training workloads that intelligently aligns computational resources (GPU, CPU) with user-defined SLA preferences through adaptive multi-objective reinforcement learning.

A. System Overview

Figure 1 illustrates the SLA-MORL architecture, and the complete process follows Algorithm 1, which operates in four phases:

Input Phase. The system accepts two inputs: (1) a model-data pair consisting of ML training code and dataset, and (2) user-defined SLA preferences specified as either `time` priority (e.g., complete within 60 minutes), `cost` priority (e.g., spend less than $20), or `balanced` (e.g., optimize both within given targets).

Offline Phase (Optional). Before optimization begins, SLA-MORL can leverage prior knowledge through three possible paths: (1) load historical CSV logs and extract patterns to pre-train the Actor-Critic networks, (2) run baseline experiments using 20% of data for 10% of epochs to gather initial performance estimates, or (3) skip this phase entirely and proceed directly to online optimization.

Online Phase. The core optimization loop executes continuously throughout model training with five interacting components. The **State Vector** constructs a 21-dimensional representation of system status. The **Change Detector** monitors for significant changes or SLA violations. The **Actor-Critic Network** makes resource allocation decisions. The **Execute** function implements resource changes through system APIs. The **Adaptive Reward** function computes feedback signals that dynamically adjust based on SLA compliance, forming a closed-loop system.

Output Phase. The system produces comprehensive reports including final training metrics, optimal resource configurations, SLA compliance status, and recommendations.

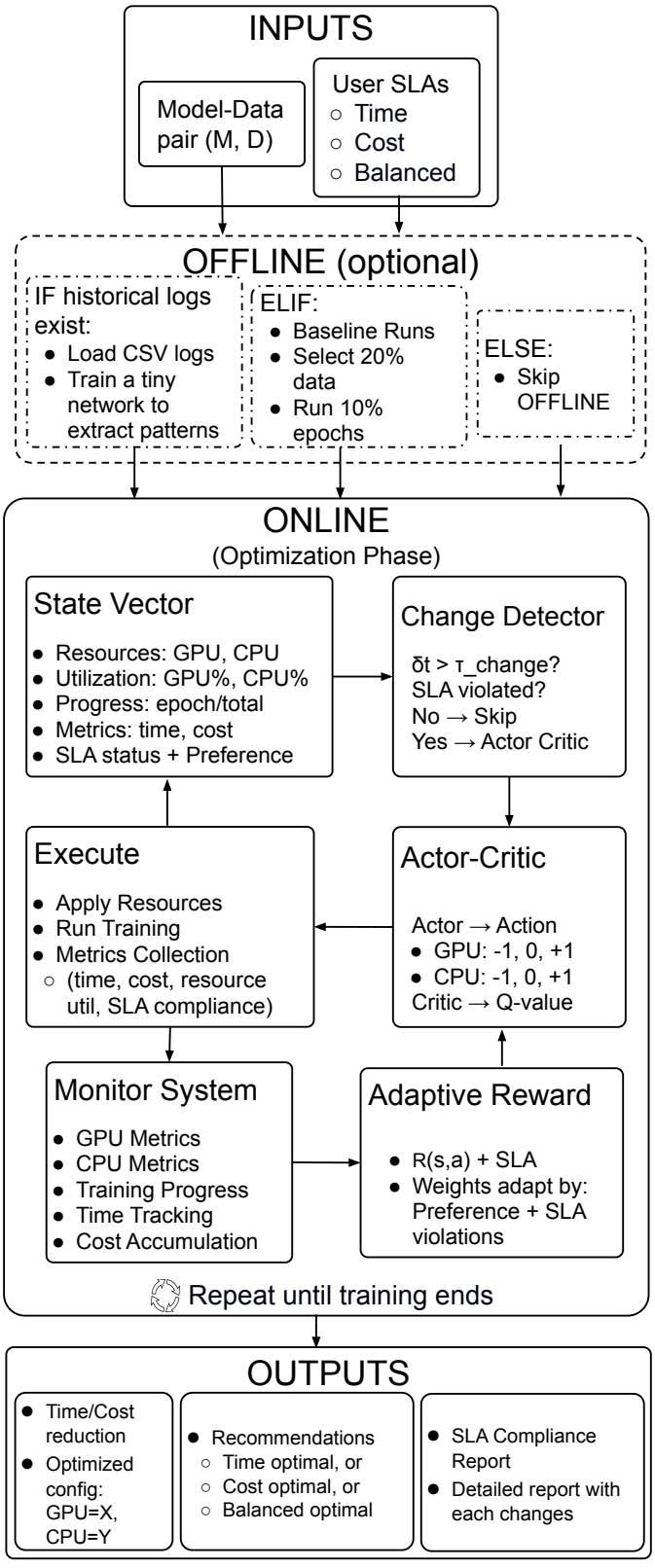

Fig. 1: Proposed SLA-MORL Architecture.

Algorithm 1 SLA-MORL Overall Training Process

Input: Model-data pair (M, D), user preference $p \in \{time, cost, balanced\}$
Output: Optimized resource allocation (g^*, c^*) and performance report
Initialize system parameters: $\tau_{change} = 0.1$, $\alpha = 0.5$, $\gamma = 0.95$
Execute Phase 1: Historical Learning or Baseline Initialization (Algorithm 3)
Execute Phase 2: Online Optimization with trained/initialized policy (Algorithm 2)
Generate Pareto front from execution history
Select best configuration (g^*, c^*) based on user preference p
Return performance report with SLA compliance analysis

B. MDP Formulation for SLA-Aware Resource Allocation

We formalize the dynamic resource allocation problem as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$.

1) State Space Design: The state vector $\mathbf{s}_t \in \mathbb{R}^{21}$ captures system dynamics through six components:

$$\mathbf{s}_t = [\mathbf{r}_t, \mathbf{u}_t, \mathbf{p}_t, \mathbf{c}_t, \mathbf{v}_t, \mathbf{w}_t], \tag{2}$$

where **Resource Allocation** $\mathbf{r}_t = [g_t, c_t]$ represents current GPU count and CPU cores; **Resource Utilization** $\mathbf{u}_t = [u_t^{gpu}, u_t^{cpu}]$ indicates normalized usage rates; **Progress Indicators** $\mathbf{p}_t = [p_t^{epoch}, p_t^{throughput}]$ tracks training progress; **SLA Compliance Flags** $\mathbf{c}_t \in \{0,1\}^6$ monitors constraint satisfaction for each preference type; **Violation Severity** $\mathbf{v}_t \in [0,1]^6$ quantifies violation magnitude; and **User Preference** $\mathbf{w}_t \in \{0,1\}^3$ encodes the selected priority mode.

2) Action Space and Resource Models: We define a discrete action space for stable resource control:

$$\mathcal{A} = \{(\Delta g, \Delta c) : \Delta g, \Delta c \in \{-1, 0, +1\}\}, \tag{3}$$

creating 9 possible actions. Our cost and time models are:

$$C_{hourly} = g \cdot 5.0 + c \cdot 0.5, \tag{4}$$

$$T_{epoch}(g, c) = \frac{T_{base}}{\sigma(g) \cdot \rho(c)}, \tag{5}$$

where $\sigma(g) = 1 + 0.8(g-1)$ represents GPU scaling efficiency and $\rho(c) = 1 + 0.1 \log_2(c)$ captures CPU benefit.

C. Online Optimization Process

Algorithm 2 presents our online optimization loop that continuously adapts resources during training.

The change detection threshold $\tau_{change} = 0.1$ balances responsiveness with stability, triggering adaptation when state changes exceed 10% or SLA violations occur.

Algorithm 2 Phase 2: Online Optimization with SLA-Aware Adaptation

Input: Pre-trained networks π_θ, Q_ϕ, initial state $\mathbf{s}_0$
Initialize replay buffer $\mathcal{B}$ with capacity 100K
for episode $e = 1$ to 10 **do**
  for epoch $t = 1$ to T_{epochs} **do**
    Observe current state $\mathbf{s}_t$ and compute $\delta_t = \|\mathbf{s}_t - \mathbf{s}_{t-1}\|_2$
    if $\delta_t > 0.1$ or SLA violation detected **then**
      Identify violated objectives: $V_t = \{i : c_i^{(t)} = 0\}$
      Update weights using adaptive mechanism (Eq. 9)
      Select action $a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)$ with ϵ-greedy
      Execute action and wait for stabilization
      Compute reward r_t using Equation 11
      Store transition in $\mathcal{B}$ and update networks
    end if
  end for
end for
Return optimized policy and resource trajectory

D. SLA-Aware Actor-Critic Architecture

1) Network Architecture: **Actor Network** $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maps states to action probabilities: Input(21) $\to$ Linear(128) $\to$ ReLU $\to$ Linear(64) $\to$ ReLU $\to$ Linear(9) $\to$ Softmax.

Critic Network $Q_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ estimates Q-values: Input(30) $\to$ Linear(128) $\to$ ReLU $\to$ Linear(64) $\to$ ReLU $\to$ Linear(1).

Both use Adam optimizer with $\eta_\pi = 3 \times 10^{-4}$ and $\eta_Q = 1 \times 10^{-3}$.

2) Contribution 1: Intelligent Initialization: We address cold-start through two mechanisms:

Historical Learning. Extract patterns from previous runs:

$$\mathcal{P}_{hist} = \{(g_i, c_i, M_i, D_i) \to (\theta_i, T_i, C_i)\}_{i=1}^{N}, \tag{6}$$

where (M_i, D_i) are model/data characteristics and (θ_i, T_i, C_i) are achieved metrics.

Baseline Initialization. When logs unavailable, run controlled experiments:

$$\hat{\theta}(g, c) = 5 \cdot \theta_{baseline}(g, c), \tag{7}$$

scaling from 20% data sample to full dataset performance.

3) Contribution 2: Adaptive Multi-Objective Reward System: Base weights reflect user preferences:

$$\mathbf{w}^{base} = \begin{cases} [0.6, 0.1, 0.3] & \text{if } p = \text{time priority} \\ [0.1, 0.6, 0.3] & \text{if } p = \text{cost priority} \\ [0.3, 0.3, 0.4] & \text{if } p = \text{balanced,} \end{cases} \tag{8}$$

Dynamic adaptation based on violations:

$$w_i^{adapted} = w_i^{base} + \alpha \cdot v_i \cdot \mathbb{I}_{violated}(i), \tag{9}$$

$$w_i^{final} = \frac{w_i^{adapted}}{\sum_j w_j^{adapted}}, \tag{10}$$

Algorithm 3 Phase 1: Historical Learning and Initialization

Input: Model M, Dataset D, Historical logs $\mathcal{L}$ (optional)
if Skip offline = True **then**
  Initialize uniformly: $(g_0, c_0) = (1, 2)$, $\epsilon = 0.3$
else if $\mathcal{L}$ exists **then**
  Extract patterns and pre-train Q_ϕ
  Set best historical config, $\epsilon = 0.1$
else
  Run baseline on 3 configs: $\{(1, 1), (2, 4), (4, 8)\}$
  Initialize with estimates, $\epsilon = 0.2$
end if
Return π_θ, Q_ϕ, (g_0, c_0)

where $\alpha = 0.5$ controls adaptation strength.

The multi-objective reward function:

$$R_t = w_{time}^{final} \cdot R_{time} + w_{cost}^{final} \cdot R_{cost} + w_{util}^{final} \cdot R_{util} - P_{sla}, \tag{11}$$

where R_{time} rewards throughput improvements, R_{cost} rewards cost reductions, R_{util} encourages efficient utilization near targets (80% GPU, 70% CPU), and P_{sla} penalizes violations.

IV. Experiments

A. Experimental Setup

Infrastructure. All experiments were conducted on a SLURM-based HPC cluster equipped with NVIDIA Quadro RTX 8000 GPUs (48GB VRAM each), Intel Xeon Gold 6148 processors (40 cores @ 2.40GHz), and 384GB RAM per node. Resource monitoring was performed using pynvml for GPU metrics and psutil for CPU utilization at 1Hz frequency. Resource control was achieved through SLURM's scontrol commands and CUDA environment variables.

Implementation Details. SLA-MORL was implemented using PyTorch 1.13. The Actor-Critic networks were trained using Adam optimizer with learning rates $\eta_\pi = 3 \times 10^{-4}$ for the actor and $\eta_Q = 1 \times 10^{-3}$ for the critic. We used an experience replay buffer with 100K capacity and batch sizes ranging from 32 to 512 depending on GPU allocation.

Benchmark Tasks. We evaluated SLA-MORL on 13 model-data pairs across two categories: (1) computationally intensive remote sensing models, namely CAM [26], gWaveNet [27], and DAMA [28]; and (2) standard computer vision tasks, such as VGG16 and ResNet50/101 on CIFAR-10/100, and Transformer and ResNet101 on CIFAR-100 and ImageNet100 (ImageNet with 100 classes). All models were trained for 200 epochs without early stopping to ensure a fair comparison. The evaluation covered a range of dataset sizes and model complexities.

Baseline Methods. We compared against five approaches: (1) *Basic*: fixed resource allocation (one GPU and its available CPU cores) without optimization, (2) *Static_recom*: one-time optimal allocation based on workload analysis, (3) *SLA-MORL_lite*: our method without offline phase and actor-critic part, (4) *SLA-MORL_base_runs*: using only three baseline initializations with 10% epochs and 20% randomly chosen data, and (5) *SLA-MORL_w_target_logs*: leveraging logs from similar workloads.

B. Results and Analysis

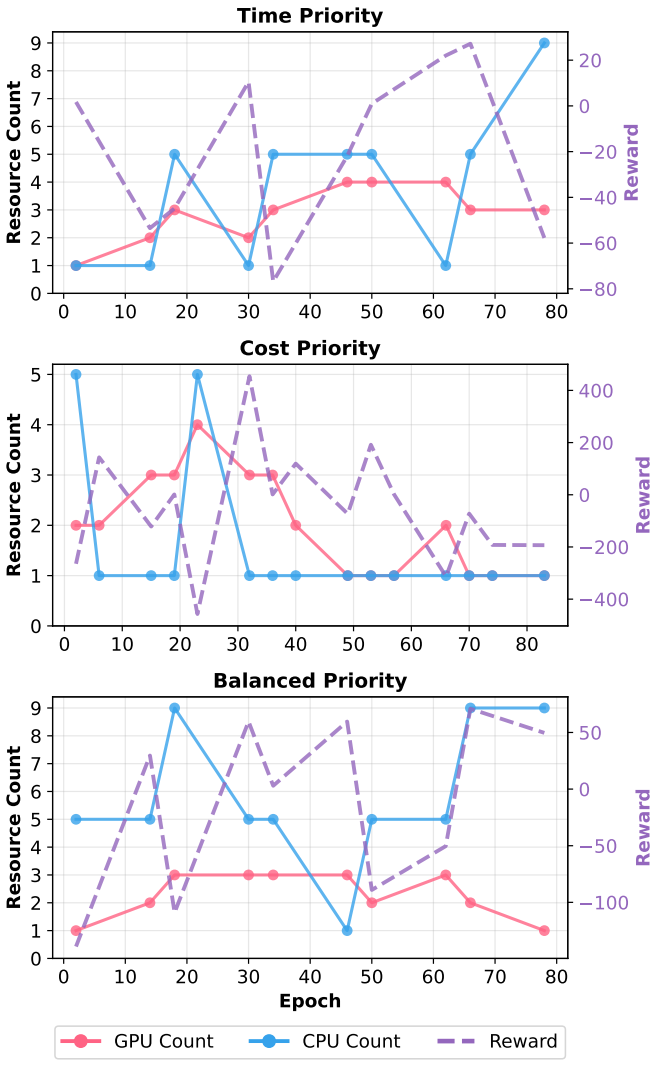

Fig. 2: Resource allocation patterns across SLA priorities showing GPU/CPU allocation and reward evolution over training epochs.

Dynamic Resource Allocation. Figure 2 demonstrates SLA-MORL's adaptive behavior across three priority configurations. When time priority is selected and jobs must complete by deadline, SLA-MORL aggressively allocates resources, scaling up to 4 GPUs when necessary to minimize training duration. This results in higher costs but ensures deadline compliance. For cost priority workloads where budget is the primary concern, the system maintains minimal resource allocation (typically 1-2 GPUs and 2-4 CPUs) to reduce operational expenses while accepting extended training times. The balanced priority exhibits the most dynamic behavior, with frequent resource adjustments as the system continuously navigates the time-cost tradeoff to satisfy both objectives

simultaneously. The reward signal validates our design: it becomes negative when resources are over-provisioned (indicating waste), and positive when resources are optimally reduced without compromising performance.

TABLE I: Performance Improvement by Priority Type (%) - Average Results Across 13 Model-Data Pairs.

Methods	Time	Cost	Balanced
Basic	-	-	-
Static_recom	33.4	38.7	39.2
SLA-MORL_lite	47.6	43.1	44.8
SLA-MORL_base_runs	58.2	46.9	52.7
SLA-MORL_w_target_logs	61.4	48.3	56.1
SLA-MORL	**63.7**	**49.8**	**58.4**

Overall Performance. Table I presents average performance improvements across all 13 model-data pairs compared to the basic baseline. The Basic method shows no values as we intentionally use it as the reference point (0%) for calculating relative improvements of all other methods. SLA-MORL achieves remarkable gains: 63.7% reduction in training time for time-priority workloads and 49.8% cost reduction for cost-priority jobs. These improvements significantly outperform static approaches, which achieve only 33.4% and 38.7% respectively. The progression from SLA-MORL_lite to the full version validates our contributions: historical learning capability adds approximately 15% additional improvement, demonstrating the value of leveraging past execution patterns.

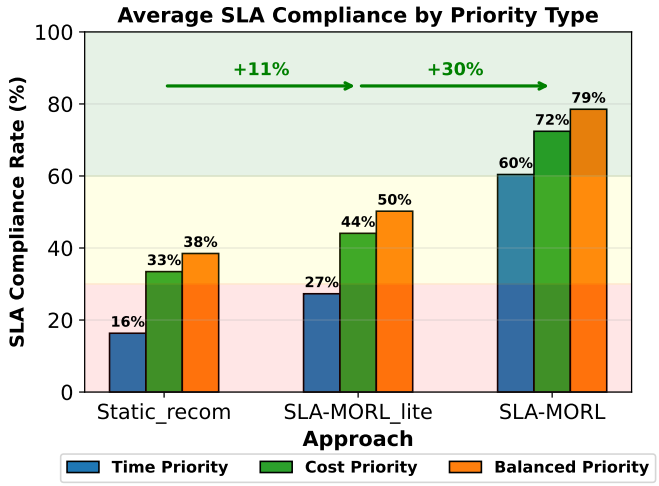

Fig. 3: SLA compliance rates showing dramatic reduction in violations across 13 model-data pairs for different priority types.

SLA Compliance Analysis. Figure 3 presents one of our most important results: the dramatic reduction in SLA violations achieved by SLA-MORL across all 13 model-data pairs. This figure demonstrates how our adaptive approach significantly improves the system's ability to meet user-defined constraints. SLA-MORL achieves 61.8% compliance rate for time priority workloads (compared to only 17.4% for static approaches), 77.1% for cost priority (versus 33.8% static), and 82.1% for balanced priority (versus 39.2% static). The consistently higher compliance rates across all priority types validate our adaptive weight mechanism's effectiveness. Time constraints prove most challenging due to their strict nature and the unpredictability of training dynamics, while balanced mode achieves the highest compliance by allowing flexible tradeoffs between objectives.

TABLE II: Comprehensive Performance Metrics Averaged Across All 13 Model-Data Pairs.

Methods	Reductions (%)		Improvements (%)	
	Time	Cost	SLA	Efficiency
Basic	-	-	19.2	-
Static_recom	48.7	50.2	59.1	49.5
SLA-MORL_lite	52.4	54.1	62.8	53.3
SLA-MORL	**67.2**	**68.8**	**73.4**	**68.0**

Comprehensive Evaluation. Table II summarizes performance across all experimental configurations. We excluded SLA-MORL_base_runs and SLA-MORL_w_target_logs as they have nearly identical results to SLA-MORL. SLA-MORL achieves 68.0% overall efficiency (calculated as the average of time and cost reductions), representing a 37% relative improvement over static approaches. The consistent gains across diverse workloads validate the generalizability of our approach.RetryClaude can make mistakes. Please double-check responses.

Pareto Frontier Analysis. Figure 4 illustrates the fundamental time-cost tradeoff across different priority configurations. The distinct regions show how each priority mode operates: time priority (blue region) accepts higher costs to achieve faster execution, cost priority (green region) minimizes expenses while tolerating longer training duration, and balanced priority (orange region) finds optimal middle ground between these extremes. SLA-MORL solutions (marked with 'stars') consistently achieve superior positions on the Pareto frontier compared to baseline methods, demonstrating our framework's ability to find better tradeoffs regardless of user preference.

Qualitative Comparison with Existing Work. Table III provides a comprehensive comparison of SLA-MORL against state-of-the-art approaches across multiple dimensions. Our framework uniquely combines several critical features absent in existing work: dynamic weight adaptation based on SLA violations, explicit user preference support, and validation on real HPC infrastructure rather than simulations. While MARS achieves 5-60% improvements on specific workflow types, SLA-MORL consistently delivers 67-69% gains across diverse ML workloads, demonstrating superior generalization.

C. Ablation Studies

We included ablation studies through SLA-MORL variations such as, SLA-MORL_lite, SLA-MORL_w_target_logs and SLA-MORL_base_runs to validate our design choices:

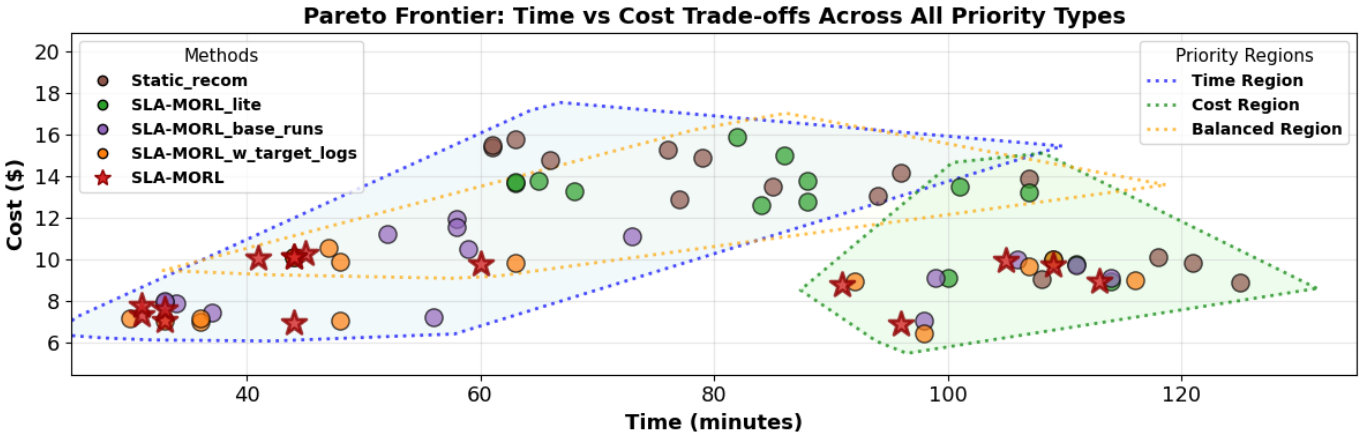

Fig. 4: Pareto frontiers for different priority configurations on gWaveNet [27] workload. Stars indicate SLA-MORL's superior solutions. Pareto frontiers for other datasets can be found in the open source link.

TABLE III: Qualitative Comparison with State-of-the-Art Resource Optimization Approaches.

Features	**Baheri et al. [10]**	**Chen et al. [7]**	**Peng at al. [25]**	**SLA-MORL (Ours)**
Optimization Objective	Cost-aware scheduling	QoS + Energy efficiency	Makespan + AWT	Multi-objective SLA
RL Algorithm	Actor-Critic ensemble	Actor-Critic DRL	Q-Learning	MORL + Actor-Critic
SLA Awareness	✓	×	✓	✓
User Preference Support	×	×	×	✓
Adaptive Weight Mechanism	×	×	×	✓
Historical Learning	Pre-trained models	×	×	✓
Performance Improvement	5-60% vs baselines	Superior QoS	Reduced makespan	67% time, 69% cost
Evaluation Environment	Workflow traces	Datacenter sim.	CloudSim simulation	Real HPC system

State Space Components. Removing SLA compliance flags ($\mathbf{c}_t$) and violation severity ($\mathbf{v}_t$) from the state representation reduces overall performance by 23%, confirming these components are essential for adaptive behavior.

Adaptive Weight Mechanism. Using fixed weights instead of our dynamic adaptation decreases SLA compliance by 31% on average, with time priority workloads suffering the most (42% drop). This validates the importance of runtime weight adjustment.

D. Limitations and Future Work

SLA-MORL is currently deployed and tested on a small scale with limited resources. Scaling to larger GPU clusters would bring additional challenges and thus require further investigation into resource management strategies. The linear cost model, while effective for current cloud pricing, may need extensions for spot instances or heterogeneous resource types. Future work includes supporting larger and multiple HPC systems, including cloud architectures such as AWS, Azure, and GCP.

V. Conclusion

This paper presented SLA-MORL, a practical multi-objective reinforcement learning framework for dynamic resource allocation in HPC environments. By addressing two fundamental challenges in adaptive resource management, we demonstrated that intelligent initialization and dynamic SLA-aware adaptation can significantly improve both efficiency and reliability of ML training workloads.

Our two key contributions, intelligent initialization to eliminate cold-start and adaptive weight adjustment for SLA violations demonstrate that dynamic resource management can significantly improve both efficiency and reliability in production ML workloads. The consistent performance gains across diverse tasks validate the practical applicability of our approach. Extensive evaluation on 13 diverse ML workloads using production HPC infrastructure validates our approach. SLA-MORL achieves 67.2% reduction in training time for deadline-critical jobs, 68.8% cost reduction for budget-constrained workloads, and 73.4% improvement in SLA compliance compared to static baselines. The consistent performance gains across different workload types and user preferences demonstrate the generalizability and robustness of our framework.

SLA-MORL represents a significant step toward autonomous cloud resource management, providing a foundation for next-generation systems that can intelligently balance performance, cost, and reliability without manual intervention. The open-source release of our implementation aims to accelerate adoption and further research in adaptive resource optimization for increasingly complex computational environments.

VI. Acknowledgment

This work is supported by grant OAC–1942714 from the National Science Foundation (NSF) and grant 80NSSC21M0027 from the National Aeronautics and Space Administration (NASA).

References

[1] R. S. Sutton, A. G. Barto *et al.*, "Reinforcement learning: An introduction," 1998.

[2] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1577–1594.

[3] Y. Gu, Z. Liu, S. Dai, C. Liu, Y. Wang, S. Wang, G. Theodoropoulos, and L. Cheng, "Deep reinforcement learning for job scheduling and resource management in cloud computing: An algorithm-level review," *arXiv preprint arXiv:2501.01007*, 2025.

[4] J. Jimenez, P. Soto, D. De Vleeschauwer, C.-Y. Chang, Y. De Bock, S. Latre, and M. Camelo, "Resource allocation of multi-user workloads in cloud and edge data-centers using reinforcement learning," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–5.

[5] W. Mao, H. Qiu, C. Wang, H. Franke, Z. Kalbarczyk, R. Iyer, and T. Basar, "A mean-field game approach to cloud resource management with function approximation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 243–36 258, 2022.

[6] D. Hortelano, I. de Miguel, R. J. D. Barroso, J. C. Aguado, N. Merayo, L. Ruiz, A. Asensio, X. Masip-Bruin, P. Fernández, R. M. Lorenzo *et al.*, "A comprehensive survey on reinforcement-learning-based computation offloading techniques in edge computing systems," *Journal of Network and Computer Applications*, vol. 216, p. 103669, 2023.

[7] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, "Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911–1923, 2022.

[8] Z. Zhao, X. Shi, and M. Shang, "Performance and cost-aware task scheduling via deep reinforcement learning in cloud environment," in *International Conference on Service-Oriented Computing*. Springer, 2022, pp. 600–615.

[9] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 372–382.

[10] B. Baheri, J. Tronge, B. Fang, A. Li, V. Chaudhary, and Q. Guan, "Mars: Malleable actor-critic reinforcement learning scheduler," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2022, pp. 217–226.

[11] H. Zheng, X. Luo, P. Wei, X. Song, D. Li, and J. Jiang, "Adaptive policy learning for offline-to-online reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 372–11 380.

[12] S. Wang, Q. Yang, J. Gao, M. Lin, H. Chen, L. Wu, N. Jia, S. Song, and G. Huang, "Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 081–47 104, 2023.

[13] H. Niu, Y. Qiu, M. Li, G. Zhou, J. Hu, X. Zhan *et al.*, "When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 599–36 612, 2022.

[14] Z. Liu, Q. Lin, C. Yu, X. Wu, Y. Liang, D. Li, and X. Ding, "Offline multi-agent reinforcement learning via in-sample sequential policy optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 18, 2025, pp. 19 068–19 076.

[15] N. D. Nguyen, T. T. Nguyen, P. Vamplew, R. Dazeley, and S. Nahavandi, "A prioritized objective actor-critic method for deep reinforcement learning," *Neural Computing and Applications*, vol. 33, pp. 10 335–10 349, 2021.

[16] E. Liu, Y.-C. Wu, X. Huang, C. Gao, R.-J. Wang, K. Xue, and C. Qian, "Pareto set learning for multi-objective reinforcement learning," *in AAAI*, 2025.

[17] X.-Q. Cai, P. Zhang, L. Zhao, J. Bian, M. Sugiyama, and A. Llorens, "Distributional pareto-optimal multi-objective reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 15 593–15 613, 2023.

[18] A. Rakshit, S. Reddy, R. Ramnath, A. Arora, and J. Boubin, "Righteous: Automatic right-sizing for complex edge deployments," in *2024 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2024, pp. 15–28.

[19] Z. Zhou, M. Huang, F. Pan, J. He, X. Ao, D. Tu, and Q. He, "Gradient-adaptive pareto optimization for constrained reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 443–11 451.

[20] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "A multi-objective reinforcement learning algorithm for deadline constrained scientific workflow scheduling in clouds," *Frontiers of Computer Science*, vol. 15, pp. 1–12, 2021.

[21] A. Souza, K. Pelckmans, and J. Tordsson, "A hpc co-scheduler with reinforcement learning," in *Job Scheduling Strategies for Parallel Processing: 24th International Workshop, JSSPP 2021, Virtual Event, May 21, 2021, Revised Selected Papers 24*. Springer, 2021, pp. 126–148.

[22] P. Haratian, F. Safi-Esfahani, L. Salimian, and A. Nabiollahi, "An adaptive and fuzzy resource management approach in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 907–920, 2017.

[23] J. Dai, J. Ji, L. Yang, Q. Zheng, and G. Pan, "Augmented proximal policy optimization for safe reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7288–7295.

[24] L. Ran, X. Shi, and M. Shang, "Slas-aware online task scheduling based on deep reinforcement learning method in cloud environment," in *2019 IEEE 21st international conference on high performance computing and communications; IEEE 17th international conference on smart city; IEEE 5th international conference on data science and systems (HPCC/SmartCity/DSS)*. IEEE, 2019, pp. 1518–1525.

[25] Z. Peng, D. Cui, Y. Ma, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for cloud computing under sla constraint," in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2016, pp. 142–147.

[26] Z. H. Tushar, A. Ademakinwa, J. Wang, Z. Zhang, and S. Purushotham, "Joint retrieval of cloud properties using attention-based deep learning models," *arXiv preprint arXiv:2504.03133*, 2025.

[27] S. A. M. Mostafa, O. Faruque, C. Wang, J. Yue, S. Purushotham, and J. Wang, "gwavenet: Classification of gravity waves from noisy satellite data using custom kernel integrated deep learning method," in *International Conference on Pattern Recognition*. Springer, 2025, pp. 164–180.

[28] X. Huang, C. Wang, S. Purushotham, and J. Wang, "Vdam: Vae based domain adaptation for cloud property retrieval from multi-satellite data," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 2022, pp. 1–10.