Energy-Aware Distributed Data-Center GPU Scheduling via Constrained Hybrid SAC with Adaptive Frequency

Ma Viet Duc, Nguyen Dang Duong, Nguyen Duc Huy, Nguyen Duc Tuyen,
Nguyen Tai Hung, Nguyen Huu Thanh
Hanoi University of Science and Technology, Hanoi, Vietnam

***Abstract*—AI training jobs increasingly arrive from multiple ingress points and must be placed across heterogeneous, geo-distributed GPU data centers. In this scenario, energy use increases rapidly if schedulers overlook where to route, how many GPUs to allocate, and which frequency to use. We present CHSAC–AF (Constrained Hybrid Soft Actor–Critic with Adaptive Frequency), a learning-based scheduler that factorizes the discrete action into data-center choice and GPU-count selection with feasibility masks, and delegates per-site DVFS to a model-based oracle that picks the energy-optimal clock on the local discrete frequency set. Constraints on queue/latency are handled via a CMDP with dual updates, and distributional critics stabilize learning under heavy-tailed service times. In a WAN-connected simulation with heterogeneous GPU clusters, CHSAC–AF simultaneously improves capacity and efficiency, completing jobs up to 1.6 times faster than a default policy while reducing the average energy per unit of work by about 17% compared to the default baseline. The learned policy keeps queues stable by steering arrivals away from bottleneck sites and selecting just-enough GPUs with energy-appropriate frequencies, demonstrating that energy-aware scheduling need not sacrifice throughput or service quality in distributed GPU data centers.**

***Index Terms*—Distributed Data Center, Large Language Model, Fine-tuning Model, Reinforcement Learning, Cloud Computing**

I. Introduction

Large-scale Artificial Intelligence (AI) development has shifted from a handful of centrally hosted training runs to a steady stream of fine-tuning jobs from many tenants, enterprises adapting foundation models to domains, product teams iterating on features, and researchers exploring new prompts and adapters. These workloads are increasingly launched from multiple geographic ingress points and must be placed onto distributed Data Centers (DCs) that differ in GPU types, counts, queues, and network positions. In this setting, routing jobs to the "right" site and allocating just-enough accelerators are as important as raw cluster scale for sustaining throughput and user experience.

Energy and carbon are now primary considerations in DC, particularly in geographically distributed AI training. A bottom-up analysis in [1] estimates that data centres consumed 205 TWh in 2018 ($\approx 1\%$ of global electricity), establishing the scale of the baseline DC load. Building on this baseline, worldwide AI workloads alone could add 85–134 TWh per year by 2027 if current hardware deployment trends continue, underscoring the need for energy-aware routing, GPU allocation, as well as methods for dynamic voltage and frequency scaling (DVFS) [2].

Compounding the challenge, operators face hard physical limits in power delivery, cooling, and space at single sites. The industry response is to "scale across": distribute AI workloads over multiple data centers and interconnect domains rather than attempting to scale further within one campus. In such geo-distributed deployments, admission control, routing, GPU-count selection, and DVFS become tightly coupled decisions that determine both energy efficiency and service quality.

As illustrated in Fig. 1, we consider a distributed DC environment with multiple ingress nodes acting as request traffic from users. Each ingress receives training/fine-tuning requests that must be admitted, routed to a data center, and assigned a GPU count; the chosen DC then runs the job at an appropriate GPU frequency. The objective is to minimize energy per unit of useful work while satisfying system-level service constraints (e.g., keeping queues stable and meeting latency/throughput targets). The key difficulty is balancing heterogeneous, time-varying capacities across sites with the energy–performance trade-offs induced by GPU counts and DVFS.

In this paper, we introduce *Constrained Hybrid Soft Actor–Critic with Adaptive Frequency* (CHSAC–AF), a learning-based scheduler that (i) learns capacity-aware routing and GPU-count selection via a masked, factorized categorical policy over discrete DC and GPU actions, while (ii) delegating per-site GPU frequency to a model-based oracle that selects the energy-optimal frequency on the local discrete DVFS set. Constraints such as latency/queue budgets are handled through a Constrained Markov Decision Process (CMDP) formulation with dual updates, allowing the policy to be explicitly energy-aware without sacrificing service guarantees. This hybrid design shrinks the exploration space, respects feasibility at training time, and adapts naturally to heterogeneous, geo-distributed clusters.

Our contributions are listed as follows:

- We propose CHSAC–AF, a hybrid reinforcement learning method that factorizes the discrete action space with feasibility masks and uses a model-based energy-optimal frequency oracle to minimize per-site energy.

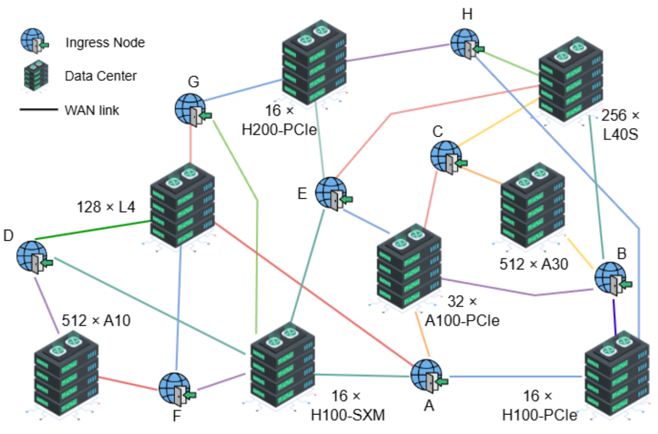

Fig. 1. Distributed Geographical Data Center architecture with multi-ingress nodes for AI training (fine-tuning LLM) requests.

- We incorporate constraint handling via dual variables and distributional critics to stabilize learning under heavy-tailed service times and dynamic feasibility.
- We build a simulator of heterogeneous, WAN-connected DCs and show that CHSAC–AF increases completed jobs while reducing energy per unit, demonstrating that energy awareness need not come at the expense of capacity.

II. Related Work

Motivated by the growing environmental concerns, recent studies have shifted focus toward carbon- and energy-aware scheduling strategies. At the cluster level, [3], [4] introduce power/performance models as functions of GPU count and clock frequency, thereby enabling energy-efficient resource allocation for heavy AI workloads. For LLM serving, Stojkovic et al. [5] adaptively tune instance count, parallelism, and GPU frequency to reduce energy and carbon under latency constraints. Exploiting temporal flexibility, Kang et al. [6] follow intraday electricity prices to cut cost under deadlines, whereas [7] shift workloads to low-carbon periods with minimal job completion time (JCT) impact. The above systems provide solid foundations for energy-efficient cluster management, but they are typically single-cluster and lack scalability to distributed deployments.

A few studies extend scheduling to geo-distributed and edge–cloud settings, emphasizing cross-site coordination and renewable integration [8]. Some efforts leverage RL for distributed scheduling: Xing et al. [9] decouple job ordering from placement to reduce JCT, and Sarkar et al. [10] propose hierarchical, carbon-aware spatio-temporal control. Still, existing methods suggest that energy-aware scheduling in distributed settings remains underexplored.

To bridge this gap, we present CHSAC–AF, a constrained hybrid SAC scheduler for geo-distributed data centers. It learns to couple cross-site routing and GPU-count selection with a model-based DVFS oracle. We enforce latency/queue budgets with a CMDP, yielding energy-aware decisions that preserve service guarantees across heterogeneous sites.

III. System and Modeling

We consider a geo-distributed GPU infrastructure with ingress set $\mathcal{I}$ and data-center set $\mathcal{D}$. Each job j arriving at ingress $i \in \mathcal{I}$ has a training work size S_j (in units). If j is routed to $d \in \mathcal{D}$, allocated n GPUs, and the site runs at a GPU frequency $f \in \mathcal{F}_d$, the per-unit service time and active power are $T_{\text{unit}}(n, f; d)$ and $P(n, f; d)$, yielding per-unit energy. We then define:

$$E_{\text{unit}}(n, f; d) = P(n, f; d) T_{\text{unit}}(n, f; d) \tag{1}$$

In practice, the processing-time model for a training job and its energy-consumption model have been extensively studied in conjunction with DVFS, which captures the impact of clock frequency on performance and power [3], [11]. At event t, the scheduler chooses an action $a_t = (d_t, n_t)$; frequency is then selected by a local oracle as the energy-optimal discrete clock

$$f^{\star}(d, n) \in \arg\min_{f \in \mathcal{F}_d} E_{\text{unit}}(n, f; d) \tag{2}$$

Let U_t be the number of work units processed during step t and let D_j denote the end-to-end latency of job j. Our objective is to minimize long-run energy per useful work while preserving service quality and queue stability across sites:

$$\min_{\pi} J(\pi) = \limsup_{T \to \infty} \frac{\mathbb{E}\left[\sum_{t=1}^{T} E_{\text{unit}}\big(n_t, f^{\star}(d_t, n_t) d_t\big), U_t\right]}{\mathbb{E}\left[\sum_{t=1}^{T} U_t\right]} \tag{3}$$

$$\text{s.t. } \mathbb{E}[D_j] \leq \tau_{\text{lat}}, \overline{Q}_d \leq Q_d^{\max}, \forall d \in \mathcal{D} \tag{4}$$

Here $\limsup$ denotes the limit superior of the long-run average ratio, τ_{lat} sets a latency budget, and $\overline{Q}_d$ bounds long-run queue occupancy at each d. This formulation captures the core trade-off our system optimizes, routing and right-sizing GPU allocations to minimize energy per unit of completed work, while explicitly constraining latency and stability in a heterogeneous, geo-distributed environment.

IV. Proposed Method

We propose a constrained hybrid reinforcement-learning approach that (i) selects a data center and a GPU count via a masked categorical policy, and (ii) delegates GPU frequency to a model-based energy-optimal oracle. We next detail the CMDP formulation, policy/critic design, and the training loop in Algorithm 1.

A. Constrained Markov Decision Process formulation

Building on the long-run energy-efficiency objective in (3) under the service constraints in (4), we formulate the scheduling problem as a CMDP [12] as $(\mathcal{S}, \mathcal{A}, Pr, r_t, \{c_k\}, \{\tau_k\}, \gamma)$. At step time t:

- State $s_t \in \mathcal{S}$ with state space $\mathcal{S}$ stacks per-DC features: total/busy/free GPUs, current frequency, queue lengths for training, and time.
- Action $a_t = (d_t, n_t) \in \mathcal{A}$ with action space $\mathcal{A}$, dictates the data center selection (d_t) and GPU allocation (n_t).

- Transition probability $Pr(s_{t+1}|s_t, a_t)$ follows the environment (arrivals, service time, service completion).

In this CMDP formulation, service constraints defined by c_k and τ_k are enforced via Lagrangian relaxation, guiding the policy optimization detailed in Algorithm 1.

Let E_{job} be the job energy and U_t the number of units processed at step t. Define the per-unit energy $E_t^{\text{unit}} = \frac{E_{\text{job}}}{U_t}$. Let n_t be the number of GPUs allocated by the policy and $\alpha_n > 0$ a small coefficient that softly prefers fewer GPUs. The reward used in our implementation is

$$r_t = -E_t^{\text{unit}} + \frac{\alpha_n}{n_t} \tag{5}$$

The term $-E_t^{\text{unit}}$ promotes energy efficiency per unit of useful work, making improvements comparable across heterogeneous jobs and sites. In training workloads, using fewer GPUs typically reduces energy consumption, albeit at the cost of longer training time [13]. To balance this trade-off, we add a small preference term, α_n/n_t, to the reward. This gently discourages over-provisioning GPUs, and the coefficient α_n tunes the strength of this preference so the adjustment remains modest when marginal energy savings are negligible. Hard service requirements (e.g., latency or throughput thresholds) are not baked into r_t; instead, they are enforced via the Lagrangian penalties in (6)–(7), cleanly separating utility from constraints.

This objective treats r_t as the instantaneous utility and penalizes any violations $c_k(s_t, a_t) > \tau_k$ via the positive-part operator $(x)_+ = \max(x, 0)$, while preserving exploration through an entropy regularizer on the policy. Therefore, the Lagrangian objective maximized by the policy is:

$$\mathbb{E}_\pi\Big[\sum_{t\geq 0}\gamma^t\big(r_t - \sum_k \lambda_k\,(c_k(s_t,a_t) - \tau_k)_+ + \alpha\,\mathcal{H}(\pi(\cdot|s_t))\big)\Big] \tag{6}$$

Here, the entropy term $\alpha\mathcal{H}(\pi(\cdot|s_t))$ balances exploration and exploitation; the coefficient α is automatically tuned using SAC's entropy-adjustment mechanism during actor optimization. Dual variables update as

$$\lambda_k \leftarrow \Big[\lambda_k + \eta_\lambda \mathbb{E}(c_k - \tau_k)\Big]_+ \tag{7}$$

Dual variables λ_k increase when constraints are violated, pushing the policy away from violation-inducing actions.

State Encoder. We encode the vectorized state $s \in \mathbb{R}^d$ (concatenated per-DC features and time, optionally user preferences) with a 3-layer Multilayer Perceptron (MLP):

$$h = \phi_\psi(s) = \sigma\Big(W_3\,\sigma\big(W_2\,\sigma(W_1 s + b_1) + b_2\big) + b_3\Big) \tag{8}$$

where $\sigma(\cdot)$ is ReLU, $W_1 \in \mathbb{R}^{256\times d}$, $W_2 \in \mathbb{R}^{256\times 256}$, $W_3 \in \mathbb{R}^{256\times 256}$. The latent h is shared by the categorical actor (DC, GPU count) and the twin quantile critics. This simple ReLU MLP suffices in our experiments.

Given the state s_t and the feasibility-filtered action space $\mathcal{A}(s_t)$, we embed s_t into a latent vector h_t in event time t using the encoder from (8). This shared representation parametrizes both the factorized categorical actor over (d_t, n_t) and the twin distributional critics will be shown in (10). Importantly, feasibility masks are applied to the actor logits before softmax and are respected when forming critic targets, ensuring updates remain consistent with $\mathcal{A}(s_t)$.

B. Policy and Critic

The actor is purely categorical

$$\pi_\theta(a_t|s_t) = \pi_\theta(d_t|s_t)\pi_\theta(n_t|s_t) \tag{9}$$

We learn two quantile critics Q_ϕ^1, Q_ϕ^2, following distributional RL via quantile regression [14], over τ-quantiles of the return, improving stability under heavy-tailed latency. Given latent state h and action one-hots, critics output N_q quantiles. The *critic* loss is the quantile Huber loss:

$$\mathcal{L}_Q = \text{Huber}_{\text{quantile}}(Q_\phi^1, \text{target}) + \text{Huber}_{\text{quantile}}(Q_\phi^2, \text{target}) \tag{10}$$

with targets computed by the target critic and the current actor, following Soft Actor-Critic (SAC) [15]. Both the DC index and the GPU count are discrete and exhibit time-varying feasibility; a factorized policy from (9) mirrors this structure and enables per-head masking. The critics consume h_t concatenated with one-hot actions to predict return quantiles, which improves robustness under heavy-tailed service times while preserving precise credit assignment for each discrete choice.

SAC actor loss. Let $q_\pi = \min(Q_\phi^1, Q_\phi^2)$ averaged across quantiles. The policy loss with entropy temperature α is

$$\mathcal{L}_\pi = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta}[\alpha \log \pi_\theta(a|s) - q_\pi(s,a)] \tag{11}$$

and α is tuned toward a target entropy.

Given $a_t = (d_t, n_t)$, the system chooses $f^\star$ by solving (2) on the discrete frequency grid $\mathcal{F}_{d_t}$. This reduces the exploration burden, accelerates learning, and deterministically enforces an energy-minimizing DVFS choice that is compatible with RL's higher-level scheduling.

C. Training and Data Interface

At the completion of each job, the RL records the utilization U_t, the power consumption P, and the per-unit latency T_{unit}; these quantities are then used to compute r_t as in (5). During training, we update both the critic and the actor using the effective reward, defined in (12) as follows:

$$r_t^{\text{eff}} = r_t - \sum_k \lambda_k (c_{k,t} - \tau_k)_+ \tag{12}$$

Each training step ties the pieces together as follows: compute the effective reward r_t^{eff} in (12), update quantile critics with masked, bootstrapped targets, improve the actor by the SAC objective in (11) with automatic temperature tuning, and ascend the dual variables by (7). This loop jointly optimizes energy-aware utility while pushing constraint violations back under their RL thresholds.

Replay schema. This buffer is designed to be self-contained, preserving the full context of resource constraints at the time of data collection. Specifically, in addition to the state, action, and reward, we store feasibility masks for the

Algorithm 1 CHSAC–AF: Constrained Hybrid SAC with Adaptive Frequency

```
1: Initialize encoder with (8), actor (categorical DC/GPU), distributional twin critics, target critics, replay buffer.
2: Initialize Lagrange multipliers $\lambda_k \geq 0$.
3: for each environment step do
4:     Build state $s_t$; derive masks $\mathcal{D}_t, \mathcal{G}_t$.
5:     Sample $a_t = (d_t, n_t) \sim \pi_\theta(\cdot|s_t)$ with masks.
6:     Compute $f^\star \leftarrow \text{EnergyOptimalFreq}(d_t, n_t)$ via (2).
7:     Execute $(d_t, n_t, f^\star)$; observe $r_t$ by (5), constraint costs $c_{k,t}$, next state $s_{t+1}$, done.
8:     Store transition in replay.
9:     if buffer warm then
10:        Update Effective reward $r_t^{\text{eff}}$ via (12).
11:        Critic update via quantile Huber loss.
12:        SAC with entropy temperature tuning.
13:        Soft-update target critics.
14:        Update Dual variables $\lambda_k$ via (7).
15:    end if
16: end for
```

DC choice and the number of GPUs. This enables the actor, during off-policy training, to mask logits before the softmax so that no probability is assigned to invalid actions, while the critic computes Q-targets that are consistent with the action space feasible at the time the transition was collected. We store self-contained transitions in the replay buffer:

$$\big(s_t, d_t, n_t, r_t, s_{t+1}, \text{done}_t, \text{mask}_t^{\text{dc}}, \text{mask}_t^{\text{g}}\big) \tag{13}$$

Semantics of the symbols.

- done_t: Episode termination flag in $\{0,1\}$; when $\text{done}_t = 1$, bootstrap terms are disabled in critic targets.
- $\text{mask}_t^{\text{dc}}$: A validity mask over DC choices, $\text{mask}_t^{\text{dc}} \in \{0,1\}^{|\mathcal{D}|}$, where 0 indicate infeasible DCs at s (e.g., insufficient available GPUs, maintenance).
- mask_t^{g}: A validity mask over GPU-allocation choices, $\text{mask}_t^{\text{g}} \in \{0,1\}^{G_{\max}}$, where 0 indicate infeasible GPU counts at s_t.

Storing $\text{mask}_t^{\text{dc}}$ and mask_t^{g} with each transition (i) guarantees that off-policy updates respect action feasibility at the time the data were generated; (ii) prevents the actor from assigning probability mass to illegal actions (by masking logits before the softmax); and (iii) reduces non-stationarity in training when resource constraints vary over time.

During updates, batches sampled from the buffer apply the stored masks to the actor heads before computing the policy loss $\mathcal{L}_\pi$ from (11), and set the bootstrap multiplier to $(1 - \text{done}_t)$ when forming critic targets for $\mathcal{L}_Q$ from (10). This makes training reproducible and consistent under time-varying feasibility.

After all, the preceding sections have presented the essential components of our RL framework. To integrate these components into a unified optimization pipeline, we introduce Algorithm 1, denoted Constrained Hybrid Soft Actor-Critic with Adaptive Frequency (CHSAC–AF). CHSAC–AF implements a primal-dual policy-gradient method for the CMDP, jointly optimizing energy consumption and service capacity. The algorithm is explained as follows:

- Lines 1–2: Initialize the encoder with MLP from (8), the actor, two critics (Q_ϕ^1, Q_ϕ^2), the target critics, the replay buffer from (13), and the Lagrange multipliers (λ_k).
- Line 4: Construct the state s_t and the masks from (13) to ensure action validity (e.g., a data center that does not have enough GPUs is masked out).
- Lines 5–7: Sample an action $(d_t, n_t) \sim \pi_\theta$, select the energy-optimal frequency $f^\star$ according to (2), execute the action, and observe the reward, costs, and next state.
- Lines 8–13 (when the buffer is warm): (i) compute the adjusted reward r_t^{eff}; (ii) update the critics by minimizing $\mathcal{L}_Q$; (iii) update the actor by minimizing $\mathcal{L}_\pi$ with automatic tuning of α; (iv) soft-update the target critics; (v) update the multipliers λ_k. This block implements (10)–(11) with automatic temperature tuning and soft target updates

TABLE I
SIMULATION PARAMETERS

Parameter	Value
GPU types	[H200-PCIe, H100-SXM, H100-PCIe, A100-PCIe, L4, L40S, A30, A10]
GPU counts	[16, 16, 16, 32, 128, 256, 512, 512]
Arrivals	Poisson ($\lambda = 0.02$)
Job sizes	LogNormal ($\mu = 10.82$, $\sigma = 0.4$)
Simulation time	7 days
α_n	0.05
Replay buffer capacity	200000
Batch size	256

V. EXPERIMENT SETUP

A. Simulation Setup

Simulation settings. We built a simulator in Python for scheduling across geo-distributed data centers. The environment models a WAN-connected topology with 8 data centers and 8 ingress points (Fig. 1). Table I details the specific GPU types and counts per site. At each ingress, arrivals are independently generated and follow a Poisson process, and job sizes are LogNormal distributed. Code and experiment scripts are available at github.com/filrg/distributed_cluster_GPUs.

RL settings. The RL agent's off-policy training is implemented in PyTorch. Key hyperparameters, including the reward function coefficient α_n, replay buffer capacity, and batch size, are listed in Table I.

B. Baseline Methods

For comparison, we implemented three baseline methods:

- Default Policy (DP): No power/frequency-tuning is applied. Upon arrival, the scheduler allocates GPUs using the default allocation policy and runs all jobs at the GPU's default frequency (typically the maximum).

TABLE II
THE NUMBER OF COMPLETED JOBS AND THE AVERAGE ENERGY CONSUMPTION BY LOAD (E_t^{UNIT}) AFTER THE SIMULATION IS COMPLETED

Algorithm	DP	UCB1	JointNF	CHSAC–AF
Completed jobs	60595	63111	67396	**96399**
Energy by load (J/unit)	11.80	11.70	10.34	**9.77**

- UCB1: We formulate the choice of GPU frequency f as a k-armed bandit problem [16]. At each arrival, the scheduler selects f via the Upper Confidence Bound rule (UCB1). When a job completes, it updates the empirical reward based on the job's energy consumption per unit, so naturally, UCB1 will favor energy-efficient frequencies. UCB1 and its variants form classical foundations for the exploration-exploitation dilemma in online learning.
- JointNF: Inspired by [17], JointNF leverages frequency tuning for energy efficiency. When a job arrives, it performs a grid search over GPU counts and frequency levels to find the most energy-efficient configuration under the service-level objective (SLO) constraint.

The baseline methods do not specify inter-DC routing, hence we assume uniform distribution across all data centers.

VI. Results and Discussion

A. Power Consumption

Table II shows CHSAC–AF delivers both the highest throughput and the best energy efficiency. By jointly learning routing and frequency decisions, the agent directs jobs to suitable data centers given queue state and GPU availability, so more requests are admitted and finish. Consequently, CHSAC–AF completes 96,399 jobs – 43% higher than the next best method, JointNF (67,396). At the same time, it yields the lowest average energy per load at 9.77 J/unit, outperforming JointNF (10.34) and clearly the frequency-agnostic DP (11.80) and the frequency-only UCB1 (11.70). The resulting policy avoids congested sites, runs at optimal frequencies to maximize throughput within a fixed power budget.

Figure 2 plots the aggregate power across all DCs for four schedulers. CHSAC–AF exhibits a brief warm-up transient while probing DC/GPU-frequency combinations. It then settles at a higher mean power, as its multi-objective reward trades energy for service rate and latency by allocating more GPUs or sustaining higher utilization to keep queues short. This behavior also induces visibly larger variance as the job mix shifts. In contrast, JointNF explicitly favors smaller GPU allocations per training job, resulting in the lowest and smoothest trace with mild workload-driven oscillations. DP lacks frequency control, so its power closely tracks the incoming load and remains near the minimum—overlapping JointNF for much of the run. UCB1 explores briefly and converges to energy-efficient frequencies, producing a curve between the low-power DP/JointNF band and the higher-power CHSAC–AF regime after warm-up.

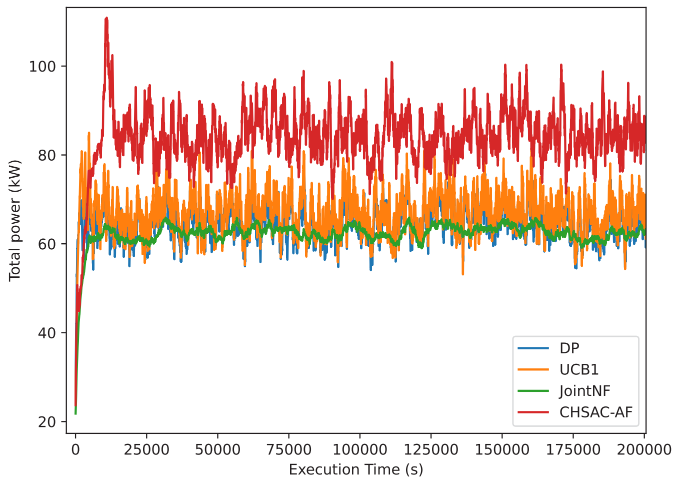

Fig. 2. Total power consumption over operating time

TABLE III
QUEUE SIZE OVER EXECUTION DAYS

Simulation time (day)	2	3	4	5	6	7
DP	9996	15166	20271	25672	30957	36083
UCB1	9401	14224	19007	24055	28868	33572
JointNF	8059	12228	16382	20712	24889	28908
CHSAC–AF	0	0	0	0	0	0

B. Training Latency

Table III shows CHSAC-–AF keeping queues essentially flat, whereas DP, UCB1, and JointNF accumulate backlog. The difference stems from capacity-aware routing: given instantaneous queues and GPU availability, CHSAC-–AF diverts arrivals from bottlenecks and avoids overload. By contrast, the baselines often send traffic to small clusters (e.g., 16×H100-PCIe/SXM, 16×H200-PCIe), where contention builds and the aggregate queue balloons. The latency distributions in Fig. 3 reflect the same trade-offs. CHSAC–AF attains lower latency than JointNF, which aggressively limits GPUs per job to save energy and therefore suffers long service times and heavy tails. DP and UCB1 show lower per-job running latency because they typically run at default or high frequencies and allocate more GPUs to each training job; however, that exhausts capacity at small-GPU DCs and increases waiting. Viewed end-to-end, CHSAC–AF keeps queues stable and maintains competitive latency, close to DP/UCB1 for running jobs, while avoiding the systemic queueing delays those methods create. This is achieved by joint routing and frequency selection that respects heterogeneous DC capacities.

C. Adaptive Routing

Figure 4 visualizes CHSAC-–AF's routing behavior on the multi-ingress, geo-distributed topology. Instead of simple nearest-site dispatch, CHSAC–AF adapts routes based on per-DC queue lengths, GPU availability, and energy state,

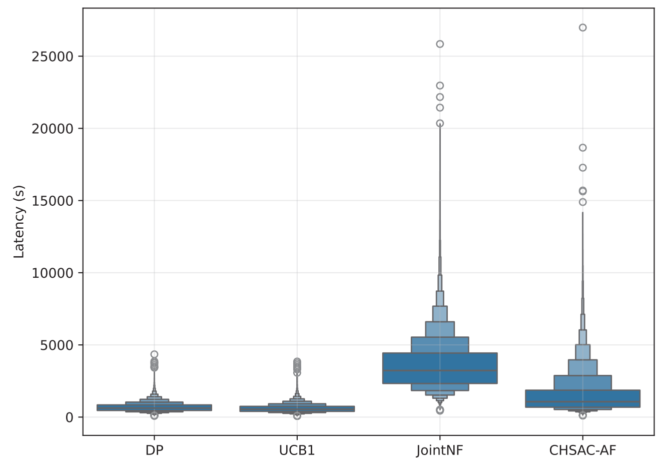

Fig. 3. Latency distributions of training jobs

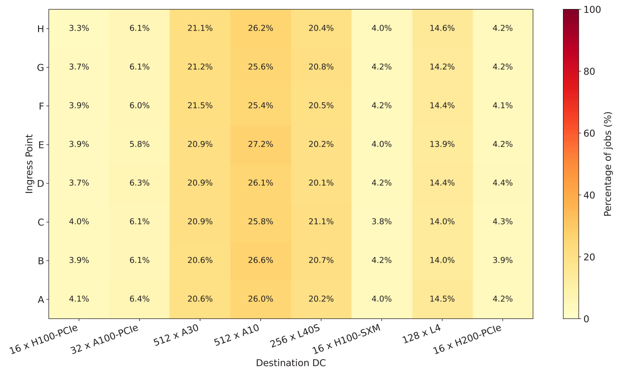

Fig. 4. Heat map of ingress-to-data-center traffic shares under the CHSAC–AF policy

favoring data centers with sparse compute while taking energy efficiency into account. Consequently, each ingress primarily dispatches requests to high-capacity clusters (512×A10/A30, 256×L40S), with only a small fraction sent to 16-GPU facilities (H100-PCIe/SXM, H200-PCIe), which are more prone to congestion. The pattern is consistent across ingress points, indicating a stable global policy that prefers compute-rich sites when feasible, uses mid-tier DCs to absorb bursts, and limits the load on small sites. Overall, CHSAC–AF balances proximity and capacity, maintaining stable performance under mixed, time-varying demand.

VII. Conclusion and Future Work

In summary, the proposed RL-based scheduler (CHSAC–AF) reduces energy per completed job while sustaining high job acceptance across heterogeneous, geo-distributed data centers. By learning from interaction rather than hand-tuned heuristics, the controller adapts automatically to diverse network topologies, hardware mixes, and traffic patterns with minimal configuration effort. A current limitation is the frequency-selection component: when thresholds are poorly calibrated, it can induce unnecessary delays.

Future work will (i) broaden large-scale parameter sweeps and ablations to characterize robustness, (ii) extend the formulation to jointly schedule inference and training workloads, and (iii) develop a fully automatic mechanism for selecting the operating frequency f that explicitly balances energy efficiency with SLA/latency targets.

Acknowledgment

This work is funded by Hanoi University of Science and Technology (HUST) under project T2025-TN-003.

References

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[2] A. De Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, 2023.

[3] D. Gu, X. Xie, G. Huang, X. Jin, and X. Liu, "Energy-efficient gpu clusters scheduling for deep learning," *arXiv preprint arXiv:2304.06381*, 2023.

[4] Y. Wang, Q. Guo, and M. Chen, "Providing load flexibility by reshaping power profiles of large language model workloads," *Advances in Applied Energy*, p. 100232, 2025.

[5] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "Dynamollm: Designing llm inference clusters for performance and energy efficiency," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1348–1362, IEEE, 2025.

[6] D.-K. Kang, K.-B. Lee, and Y.-C. Kim, "Cost efficient gpu cluster management for training and inference of deep learning," *Energies*, vol. 15, no. 2, p. 474, 2022.

[7] K. Xu, D. Sun, H. Tian, J. Zhang, and K. Chen, "{GREEN}: Carbon-efficient resource scheduling for machine learning clusters," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pp. 999–1014, 2025.

[8] Z. Miao, L. Liu, H. Nan, W. Li, X. Pan, X. Yang, M. Yu, H. Chen, and Y. Zhao, "Energy and carbon-aware distributed machine learning tasks scheduling scheme for the multi-renewable energy-based edge-cloud continuum," *Science and Technology for Energy Transition*, vol. 79, p. 82, 2024.

[9] M. Xing, H. Mao, S. Yin, L. Pan, Z. Zhang, Z. Xiao, and J. Long, "A dual-agent scheduler for distributed deep learning jobs on public cloud via reinforcement learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2776–2788, 2023.

[10] S. Sarkar, A. Guillen-Perez, V. Gundecha, A. Naug, R. L. Gutierrez, S. Mousavi, P. Faraboschi, C. Bash, and H. P. Enterprise, "Carbon-aware spatio-temporal workload distribution in cloud data center clusters using reinforcement learning,"

[11] H. Luo, K. Yang, Q. Huang, and S. Dustdar, "A novel hierarchical co-optimization framework for coordinated task scheduling and power dispatch in computing power networks," *arXiv preprint arXiv:2508.04015*, 2025.

[12] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[13] N. C. Frey, B. Li, J. McDonald, D. Zhao, M. Jones, D. Bestor, D. Tiwari, V. Gadepally, and S. Samsi, "Benchmarking resource usage for efficient distributed deep learning," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, IEEE, 2022.

[14] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, Pmlr, 2018.

[16] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[17] A. K. Kakolyris, D. Masouros, S. Xydis, and D. Soudris, "Slo-aware gpu dvfs for energy-efficient llm inference serving," *IEEE Computer Architecture Letters*, vol. 23, no. 2, pp. 150–153, 2024.