IEEE ICC 2026

JOINT PRICING AND SCHEDULING FOR SLA-AWARE SERVERLESS GPU SERVICES

Xiaoyao Huang[1], **Yanan Yang**[1], **Yujun Wang**[2], **Jie Wu**[1,3]

[1]Cloud Computing Research Institute, China Telecom [2]China Telecom Cloud Technology Co. Ltd [3]Temple University, USA

IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS

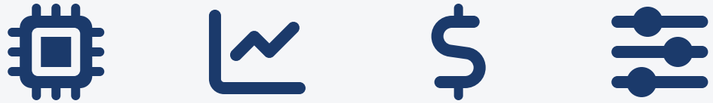
May 2026 | Glasgow, Scotland, UK

OUTLINE

Presentation overview and structure of the HOPS framework

01

Introduction & Background



Background: Serverless GPU Computing

Serverless GPU services provide elastic, pay-as-you-go GPU access for AI inference, multimodal generation, and real-time analytics.

Key characteristics:

- **Automatic scaling** — on-demand resource provisioning
- **Pay-per-use billing** — fine-grained execution-based charging
- **Function-as-a-Service** — no infrastructure management

GPU cloud market projected **$25B by 2028**, driven by GenAI adoption.

Key Observation: Demand for serverless GPU is **highly bursty and time-varying**, making capacity planning extremely difficult. Providers face a critical tradeoff between **profitability**, **resource utilization**, and **service quality**.

$25B	10-100x	<30%	ms-level
GPU Cloud Market by 2028	Demand Burstiness Peak-to-Average Ratio	Typical GPU Utilization in Serverless Settings	SLA Requirements Latency Sensitivity

Challenges & Research Motivation

Bursty Demand

Highly time-varying workloads with peak-to-average ratios of 10-100x make capacity planning extremely difficult and error-prone.

Expensive GPUs

Scarce GPU resources (A100/H100 cost $10-30K each) make even short under-utilization periods very costly for providers.

Stringent SLAs

Latency and reliability requirements with substantial financial penalties for violations. Cold-start delays can breach ms-level SLAs.

Decoupled Design

Existing methods decouple pricing from scheduling, leaving providers reactive rather than proactive in demand shaping.

Key Insight

There is a critical need to **jointly optimize pricing and scheduling** to proactively balance profitability, resource utilization, and service quality in serverless GPU platforms.

Research Gap

- **Pricing work** [13-16] focuses on VM-level or spot-instance markets — too coarse-grained for second-scale GPU elasticity
- **Scheduling work** [5-6] optimizes allocation but treats demand as exogenous — misses demand-shaping opportunity
- **No existing framework** jointly optimizes economic decisions (pricing) with system-level scheduling for serverless GPUs

02

System Model & Formulation

Two coupled timescales: window-level economic planning and slot-level real-time resource control

System Overview: Two-Timescale Architecture

Window Level (t)

Long-term economic planning

- Determines per-unit vGPU price $p_{m,t}$ for each tier m
- Announces predefined SLA s_m and baseline n^{min}
- Time scale: minutes to hours

Slot Level (h)

Real-time resource control

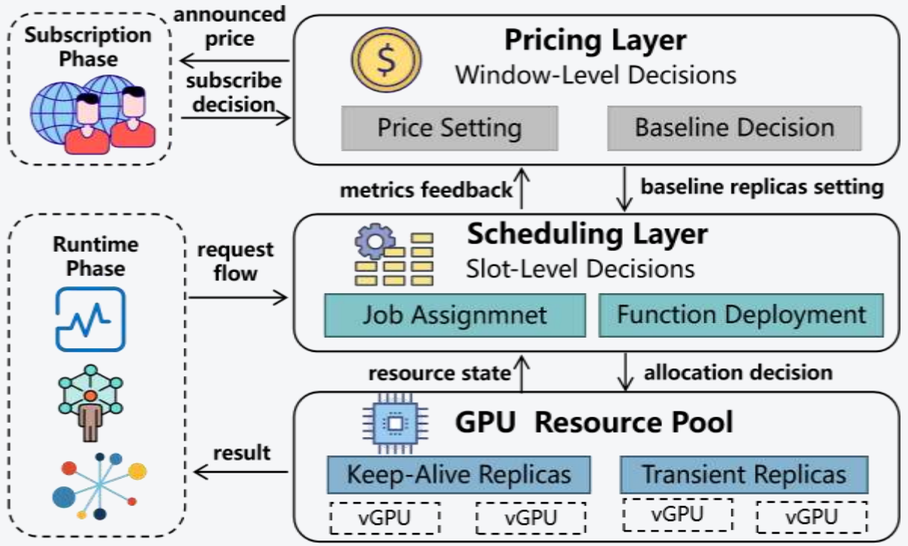
- Assigns jobs to vGPUs ($x_{k,j,h}$)
- Manages function deployment ($v_{j,f,h}$)
- Time scale: seconds

Service Tiers

Tier	Price $p_{m,t}$	SLA s_m	Baseline n^{min}
Tier 1 (Basic)	Low	Relaxed	$n^{min}_{1,t}$
Tier 2 (Standard)	Medium	Moderate	$n^{min}_{2,t}$
Tier M (Premium)	High	Strict	$n^{min}_{M,t}$

Coupling Mechanism: Pricing decisions (p_t, n^{min}_t) flow downward to constrain scheduling. Scheduling costs feed back upward to inform pricing updates. This **bi-level structure** naturally maps to a Stackelberg game formulation.

Architecture Overview



Notation: N = total vGPUs | T = number of windows | H = slots per window | M = service tiers | F = function types

Pricing Layer: Demand Model & Revenue

Multinomial logit (MNL) Demand Model

$$\lambda_{m,t} = \Lambda_t \cdot \frac{e^{\alpha s_m - \beta p_{m,t}}}{\sum_{m'} e^{\alpha s_{m'} - \beta p_{m',t}}}$$

Expected demand for tier m depends on:

- **α** — user sensitivity to SLA quality
- **β** — user sensitivity to price
- **Λ_t** — total market demand at window t

Revenue Function

$$R_t = \sum_m p_{m,t} \cdot \lambda_{m,t} \cdot g_{m,t} \cdot \Delta_t$$

Pricing Layer Objective

Maximize: Profit = Revenue − Operational Costs − SLA Penalties

Subject to: GPU capacity constraint $\Sigma_m \, n^{min}_{m,t} \leq N$

Operational Cost Components

Cost Component	Description
C_{base}	Keep-alive replica maintenance cost
C_{gpu}	Transient vGPU activation cost
C_{switch}	Function switching overhead (cold start)
$b_m L_{m,t}$	SLA penalty for tier m violations

Key Tradeoff: Higher prices → more revenue per job but lower demand. Better SLA → higher demand but more baseline cost. The pricing layer must find the optimal balance.

Scheduling Layer & Stackelberg Game

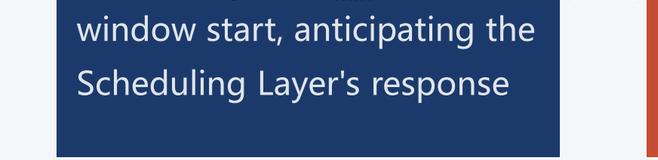
Slot-Level Decision Variables

Variable	Notation	Meaning
Job Assignment	$x_{k,j,h}$	Job k assigned to vGPU j in slot h
Function Deploy	$v_{j,f,h}$	vGPU j runs function f in slot h
Switching	$z_{j,h}$	Function switch on vGPU j

Stackelberg Game Formulation

LEADER (Pricing)

Decides p_t and $n_{min}{}^t$ at each window start, anticipating the Scheduling Layer's response



FOLLOWER (Scheduling)

Solves the slot-level MILP given upper-level decisions, minimizing operational cost + SLA penalties

Key Constraints

- **Job-function matching:** each job served by vGPU running its required function
- **One function per vGPU per slot** (no function mixing)
- **One job per vGPU per slot** (serial execution)
- **Baseline replicas:** active vGPUs ≥ $n^{min}_{m,t}$

Bi-Level Optimization

Upper Level: $\max_{pt,nmin\_t} \Phi = \Sigma_t [R_t - \Psi(p_t, n^{min}_t)]$

where $\Psi(p_t, n^{min}_t)$ = optimal objective of Scheduling Layer

Coupling: $\Psi(\cdot)$ couples economic and operational objectives — no closed form exists due to discrete scheduling decisions

03

HOPS Framework

Hierarchical Optimization for Pricing and Scheduling — coupling stochastic optimization with adaptive scheduling

PBA: Pricing & Baseline Adjustment

Challenge: Lower-level scheduling is discrete and non-linear — no closed-form gradient $\nabla\Phi$ exists. Standard gradient descent is inapplicable.

Blockwise Two-Point Gradient Estimation

Algorithm 1: PBA

Algorithm 1: Pricing and Baseline Adjustment (PBA)

Input: Initial $\mathbf{p}_0$, $\mathbf{n}_0^{\min}$, step sizes $\{\eta_t\}$, perturbation magnitude c, total horizon T.

for $t = 1, 2, \ldots, T$ **do**

- Generate random perturbation vector Δ_t with Rademacher entries;
- Compute perturbed strategies:
  $\mathbf{s}^+ = (\mathbf{p}_t + c\Delta_t^p,\ \mathbf{n}_t^{\min} + c\Delta_t^n)$,
  $\mathbf{s}^- = (\mathbf{p}_t - c\Delta_t^p,\ \mathbf{n}_t^{\min} - c\Delta_t^n)$;
- Evaluate profits Φ^+ and Φ^- by invoking Scheduling Algorithm (ARMS) under $\mathbf{s}^+$ and $\mathbf{s}^-$;
- Approximate gradient $\widehat{\nabla}\Phi_t$ using (12);
- Update variables using projected step (13);
- Pass $(\mathbf{p}_{t+1}, \mathbf{n}_{t+1}^{\min})$ to the Scheduling Layer for the next window.

Theoretical Guarantee

Theorem 1 (PBA Convergence)

With proper step sizes η_t satisfying $\Sigma\eta_t = \infty$ and $\Sigma\eta_t^2 < \infty$, PBA iterates converge **almost surely** to a stationary point of the upper-level objective Φ.

Key Properties

Gradient-Free: Only requires function evaluations, no gradient computation

Joint Update: Optimizes prices and baseline replicas together through one projected update

Dimension-Efficient: Uses only two function evaluations per iteration, independent of decision dimension *d*

ARMS: Adaptive Reconfiguration & Matching

Stage 1: Reconfiguration

Fairness-aware reconfiguration at each slot start:

1. Estimate required replicas:

$$n_{f,h}^{\text{req}} = \left\lceil (Q_{f,h} + \hat{\lambda}_{f,h}\,\Delta_h)/\mu_f\,\Delta_h \right\rceil$$

2. Compute **fairness score** across functions

3. Decide: switch keep-alive vGPU or activate transient vGPU based on **cost comparison**

Stage 2: Matching

Value- and fairness-aware matching:

1. Sort jobs by priority π_k combining:
   - Economic value (revenue + penalty)
   - SLA urgency (remaining time)
   - Fairness (deficit factor)
2. Assign to vGPU with **min predicted completion time**

Job Priority Formula

$$\pi_k = w_1 \cdot EconomicValue_k + w_2 \cdot SLAUrgency_k + w_3 \cdot FairnessDeficit_k$$

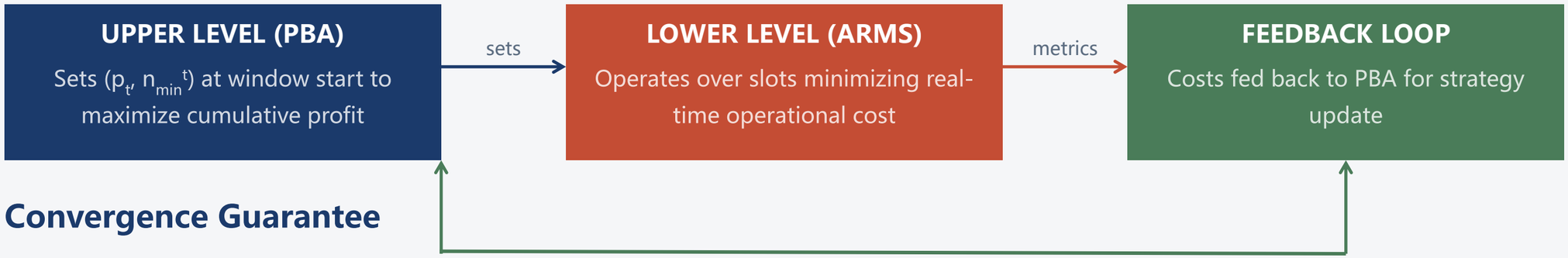
Complexity & Guarantee

Stage 1: $O(|F||J|)$ per slot | **Stage 2:** $O(|K_h| \log |K_h|)$ — efficient for online operation

Theorem 2 (ARMS Approximation)
ARMS achieves a **(1+ε)-approximation** where ε depends on workload heterogeneity. Stage 2 achieves optimal scheduling under the priority scheme.

Integrated Bi-Level Optimization

Closed-Loop Interaction



Convergence Guarantee

Theorem 3 (Bi-Level Convergence)

With PBA on a **slower timescale** (window-level) and ARMS providing **bounded-error responses** (slot-level), the joint system converges to **local Stackelberg equilibria**.

Theorems 1–3 jointly establish HOPS converges to a **stable, near-optimal equilibrium**.

Theoretical Results Summary

Theorem	Component	Result
Theorem 1	PBA (Pricing)	Converges a.s. to stationary point of upper-level objective Φ
Theorem 2	ARMS (Scheduling)	(1+ε)-approximation with efficient O(|Kh| log |Kh|) complexity
Theorem 3	Joint HOPS	Converges to local Stackelberg equilibrium with near-optimal performance

04

Performance Evaluation

Trace-driven evaluation using Azure Functions trace and Alibaba GPU cluster trace



Experimental Setup

System Configuration

Parameter	Value
Number of vGPUs (N)	64
Number of windows (T)	40
Slots per window (H)	60
Service tiers (M)	3 (Basic, Standard, Premium)
Workload traces	Azure Functions + Alibaba GPU
Function types (F)	8 diverse ML inference functions

Datasets

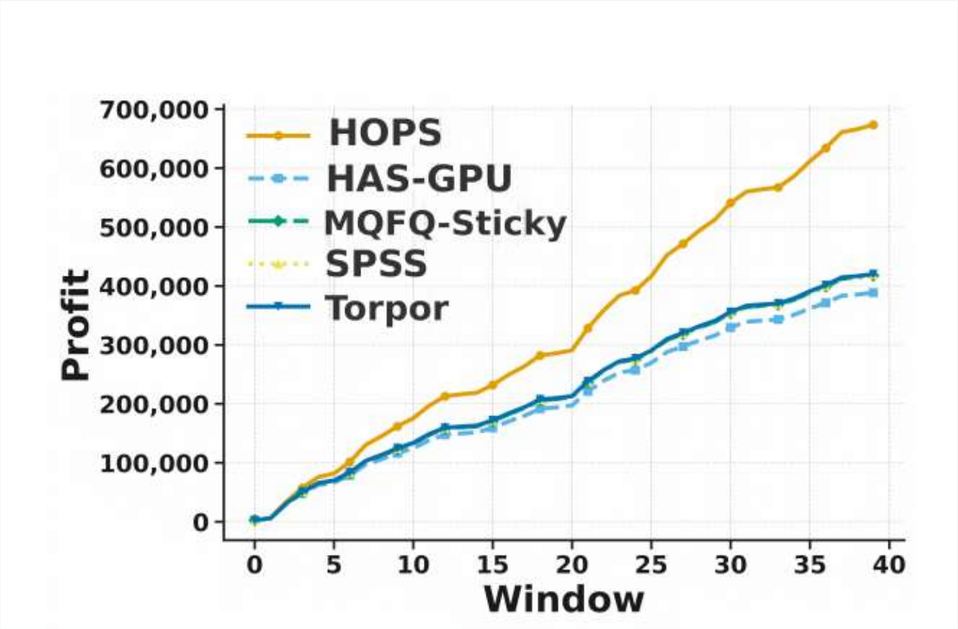
Azure Functions Trace [Shahrad et al., ATC'20]
Real-world serverless invocation patterns with bursty demand characteristics

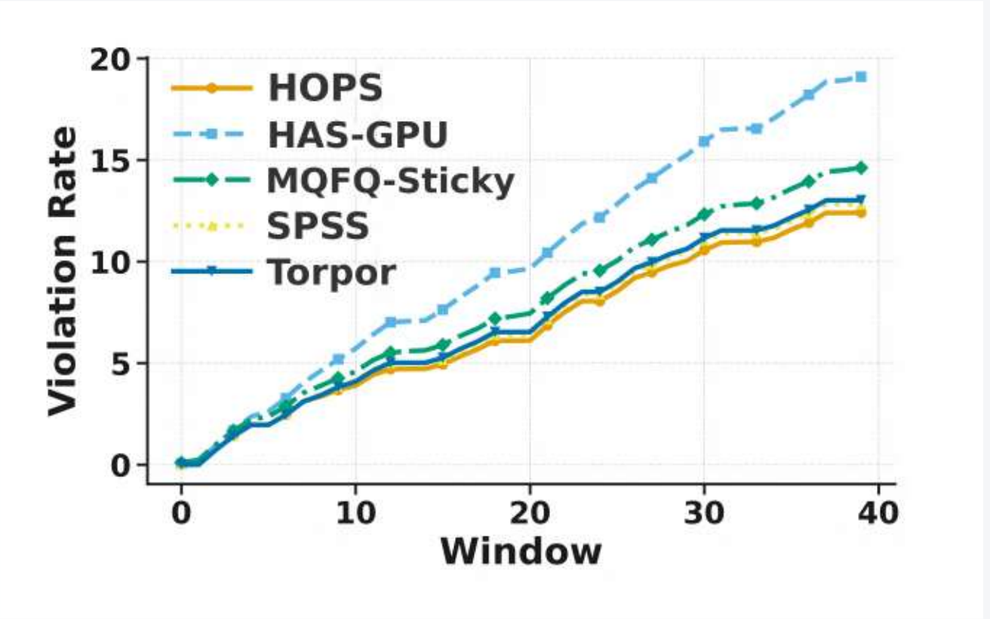
Alibaba GPU Cluster Trace [Wang et al., NSDI 2022]
Production GPU cluster traces with diverse ML workloads and resource demands

Baseline Methods

Baseline	Approach	Limitation
Torpor (ATC’ 25)	Keep-alive optimization for serverless	No pricing control; static keep-alive policies
HAS-GPU (Euro-Par’ 25)	Hybrid auto-scaling with GPU slicing	Reactive allocation only; no demand shaping via pricing
MQFQ-Sticky (arXiv’ 25)	Multi-queue fair scheduling + GPU memory management	Treats demand as exogenous; decoupled from pricing
SPSS (Ours-Abl.)	Serverless performance scaling system	No joint pricing-scheduling optimization

Profitability & SLA Compliance





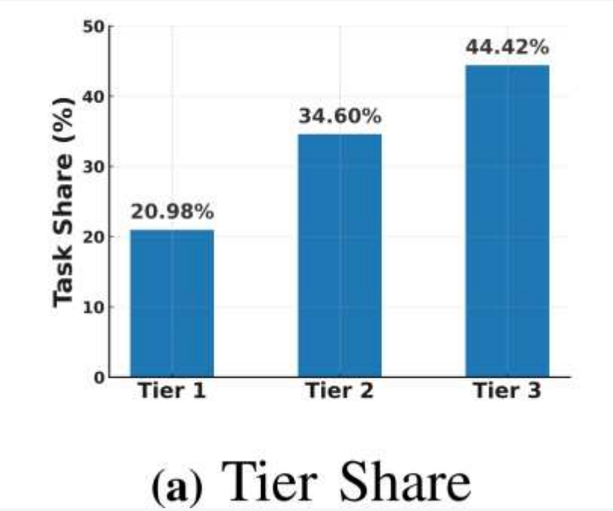
Key Findings

- HOPS achieves **45-55% profit improvement** over HAS-GPU (best baseline) by continuously adapting prices and baseline allocations
- SLA violation rate stabilizes at **~14%** vs. ~19% for HAS-GPU, demonstrating effective balance between profitability and compliance

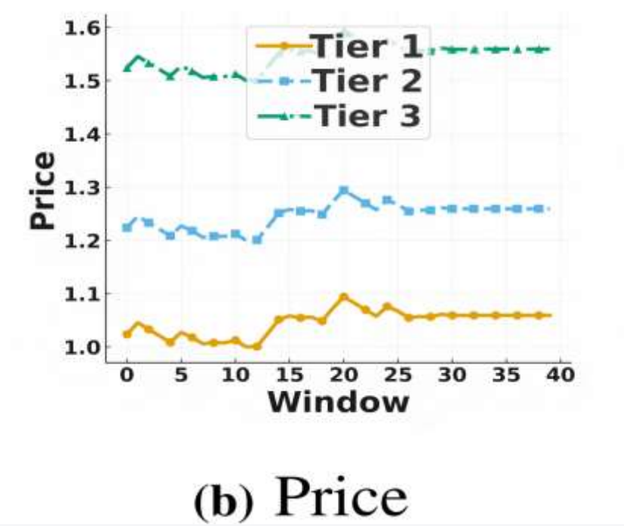
45-55%	~14%	40 windows	4 baselines
Profit Improvement	SLA Violation Rate	Convergence Period	Compared Methods

Convergence & Resource Efficiency

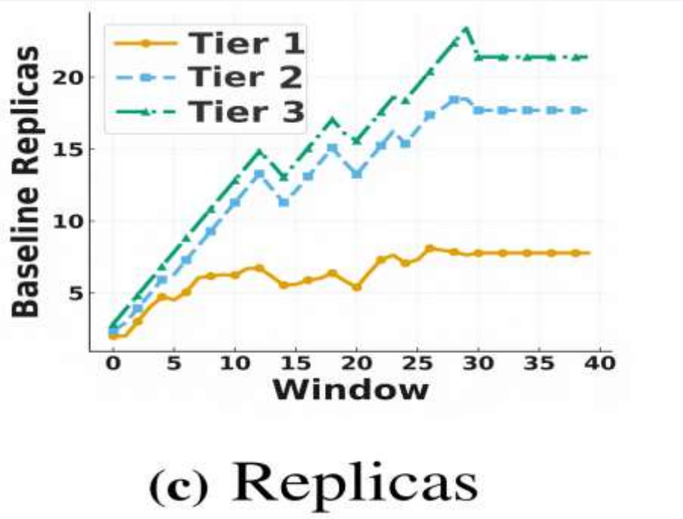
Convergence of Pricing and Baseline Decisions



(a) Tier Share

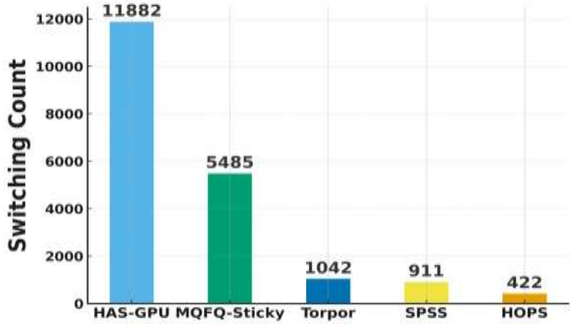


(b) Price

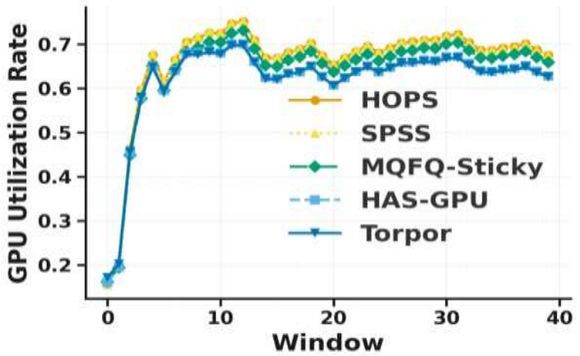


(c) Replicas

Switching Overhead Comparison



(a) Switching Overhead



(b) GPU Utilization

Key Results Summary

422 switches

≈4% of HAS-GPU (10,550), 10× lower than MQFQ

70-72% GPU utilization

Sustained without frequent reallocation

~20 windows to converge

Conclusion & Future Work

Conclusion

HOPS jointly optimizes **pricing** and **scheduling** for SLA-aware serverless GPU services via a **Stackelberg game formulation** with:

- **3 theorems** establishing convergence to near-optimal equilibrium
- **45-55% profit improvement** over state-of-the-art baselines
- **Practical design** with efficient $O(|K_h| \log |K_h|)$ complexity

Future Directions

Multi-Cluster Extension

Extend to multi-cluster and heterogeneous GPU environments (A100, H100, edge GPUs)

Online Learning

Explore online learning and meta-learning for faster convergence under dynamic workloads

Robustness

Enhance robustness under adversarial and non-stationary workload patterns

Takeaway: By integrating pricing and scheduling, HOPS transforms serverless GPU providers from **passive resource allocators** into **active demand shapers**, achieving superior profitability while maintaining SLA compliance.

THANK YOU

QUESTIONS & DISCUSSION

"Joint Pricing and Scheduling for SLA-aware Serverless GPU Services"

CONTACT

huangxy32@chinatelecom.cn | jiewu@temple.edu

Xiaoyao Huang, Yanan Yang, Yujun Wang, Jie Wu

IEEE ICC 2026