# Adaptive Fine-Tuning Scheduler for Multi-Tenant Edge LLM via Convergence-Aware Bandits

**Yandi Li**[1,3] **Jianxiong Guo**[2,3*] **Yupeng Li**[1] **Zhiqing Tang**[2] **Tian Wang**[2] **Weijia Jia**[2,3]
[1]Hong Kong Baptist University, Hong Kong
[2]Institute of AI and Future Networks, Beijing Normal University, Zhuhai, China
[3]Beijing Normal-Hong Kong Baptist University, Zhuhai, China

## Abstract

Personalizing Large Language Models (LLMs) directly on local edge servers is becoming increasingly important for privacy-preserving and context-aware applications. However, this potential is bottlenecked by hardware resources: limited GPU memory permits only a fixed number of train-ready LoRA adapters, while compute constraints enforce sequential fine-tuning updates. This creates a critical challenge: scheduling scarce update opportunities across a dynamic stream of resident tenants to improve overall model quality and scheduling stability. Crucially, standard Multi-Armed Bandit (MAB) algorithms fail to distinguish between low-potential tenant streams and fully saturated adapters, leading to wasteful exploration on tasks that offer little further gain. To address this, we propose WCA-UCB (Windowed Convergence-Aware UCB), a scheduler designed for this piecewise-converging non-stationary environment. By formally modeling fine-tuning as a slot-constrained bandit problem with piecewise-converging rewards, WCA-UCB detects when an adapter has saturated to pause its training and resets stale statistics after tenant replacement or harmful drift. We prove a dynamic regret bound of $\tilde{O}(\log T)$ and validate the system on an edge-like single-GPU LoRA prototype running Qwen2.5-1.5B. Results demonstrate that WCA-UCB reduces model perplexity by $2.6\%$ to $8.6\%$ and uses up to $5.8\times$ fewer training-target switches than strong non-stationary baselines. These results support convergence-aware scheduling as a promising approach to update allocation in resource-constrained local LLM personalization.

## 1 Introduction

Running large language models (LLMs) close to users and data sources is attractive for latency-sensitive and privacy-constrained applications [Tian et al., 2025, Xu et al., 2024, Shen et al., 2025]. Since a single generic model rarely matches heterogeneous local tasks, parameter-efficient fine-tuning (PEFT), especially LoRA-style adapters [Houlsby et al., 2019, Li and Liang, 2021, Lester et al., 2021, Hu et al., 2022, Dettmers et al., 2023, Ding et al., 2023], provides a lightweight path for personalization.

We study an emerging local-personalization workload in which an edge server mainly serves inference but periodically performs small LoRA updates for multiple tenants, such as departments, devices, applications, or local user groups. This workload is constrained by two resources: only a few adapters and optimizer states can be kept train-ready in GPU memory [Kingma and Ba, 2015, Rajbhandari et al., 2020, 2021], and the training pipeline can update only one resident adapter at a time. Thus, conditioned on the current resident set, the scheduler must decide which adapter should receive the next incremental update.

*Jianxiong Guo is the corresponding author (jianxiongguo@bnu.edu.cn).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

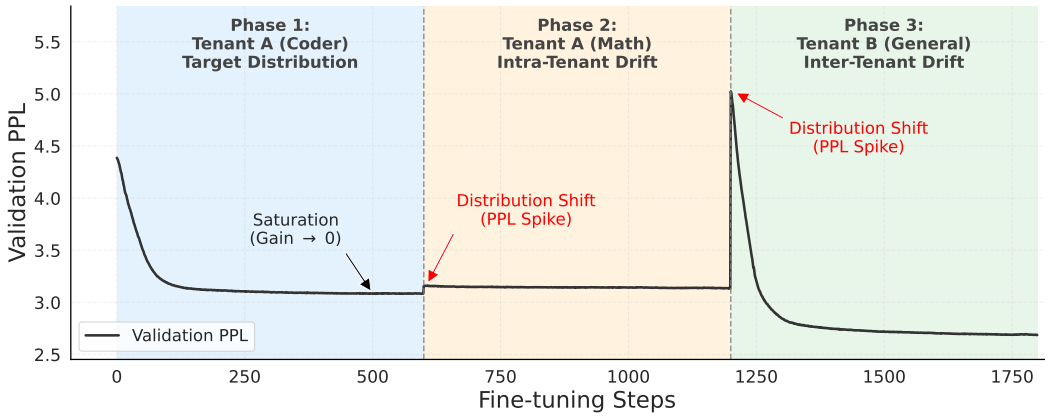



Figure 1: Illustrative validation Perplexity (PPL) dynamics during continual adapter fine-tuning. Phase 1 shows natural learning saturation: PPL drops rapidly and then plateaus, indicating diminishing marginal gains. Phases 2 and 3 illustrate harmful task or tenant shifts that cause PPL spikes and make stale histories unreliable. Explicit tenant replacement can also be handled through slot-identity changes even when the new tenant is not initially harder than the previous one. Together, these dynamics motivate the *piecewise-converging* reward abstraction.

A simple alternative is to train one tenant until convergence and then switch. This can be inefficient when active tenants have different marginal gains, sessions are short-lived, or a low-gain tenant occupies the pipeline while other resident adapters remain valuable. Our goal is not to force frequent alternation, but to decide online whether to continue updating the same adapter or reallocate compute to another resident slot. Figure 1 gives an intuitive example of why this decision is non-trivial: useful updates naturally saturate, while task or tenant shifts can make previous histories stale.

We adopt a slot-based abstraction: a slow system layer maps tenants to $K$ GPU-resident adapter slots, while a fast learning layer treats each slot as a bandit arm. Pulling an arm performs one incremental fine-tuning step on the adapter currently resident in that slot and yields a net reward based on marginal learning gain minus adaptation cost. The scheduler may select the same slot for many consecutive updates; the action granularity only allows it to revise this decision after each observed update. Figure 2 summarizes this two-tier control loop.

The resulting rewards are *piecewise-converging*. Within a stable slot segment, marginal gains decay as the adapter approaches saturation [Domhan et al., 2015, Hestness et al., 2017]; after slot repopulation or harmful drift, stale histories may become unreliable [Gama et al., 2014, Bayram et al., 2022]. This creates two coupled decisions: when to stop spending updates on a saturated adapter, and when to reset outdated statistics after a distribution shift.

Classical stochastic bandits assume stationarity [Auer et al., 2002], generic non-stationary bandits rely mainly on forgetting or change detection [Garivier and Moulines, 2008, Besbes et al., 2014, Besson et al., 2022], and rotting bandits model irreversible decay [Seznec et al., 2020]. None directly captures convergence saturation, resettable learning potential, and finite VRAM-slot constraints together. We therefore propose **WCA-UCB**, which combines windowed UCB, gain-based saturation detection, and proxy-based reset.

Our contributions are: (1) a slot-constrained bandit formulation for multi-tenant local adapter adaptation; (2) a piecewise-converging reward abstraction capturing saturation and resettable learning potential; (3) WCA-UCB, a lightweight scheduler for convergence-aware update allocation; (4) a dynamic regret bound with explicit dependence on arm-specific regime changes; and (5) trace-driven and a multi-tenant LoRA prototype evaluations showing improved perplexity and fewer adapter context switches than non-stationary bandit baselines.

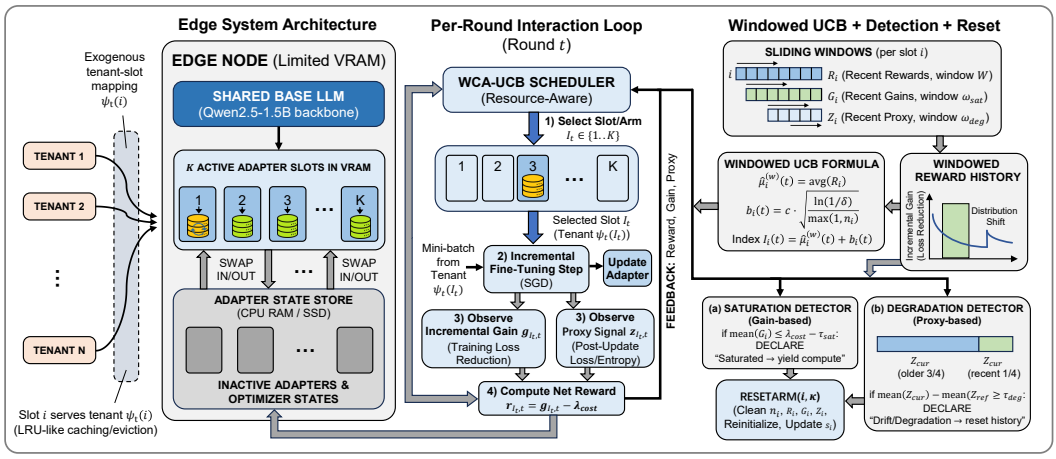



Figure 2: **Slot-based edge fine-tuning loop.** A slow system layer maps tenants to $K$ VRAM-resident adapter slots, while WCA-UCB treats resident slots as bandit arms and allocates one incremental update per round. Sliding-window reward, gain, and proxy histories support UCB selection, saturation freezing, and degradation reset.

## 2 Related Work

**PEFT and multi-tenant adapter systems.** Parameter-efficient fine-tuning adapts a frozen backbone with a small number of trainable parameters, including adapters [Houlsby et al., 2019], prompt/prefix tuning [Li and Liang, 2021, Lester et al., 2021], and LoRA [Hu et al., 2022]; quantized variants further reduce memory cost [Dettmers et al., 2023, Ding et al., 2023]. Multi-tenant LLM serving systems manage many adapters over a shared backbone using paging, batching, and caching mechanisms [Kwon et al., 2023, Chen et al., 2024, Sheng et al., 2023, Wu et al., 2024, Shen et al., 2025]. These works mainly target inference-time serving or system-level adapter management. In contrast, we study how to allocate scarce *training* updates among resident adapters whose marginal gains evolve online.

**Scheduling and bandits under changing rewards.** GPU scheduling and offloading systems improve resource multiplexing for training workloads [Xiao et al., 2018, Gu et al., 2019, Yu and Chowdhury, 2020, Rajbhandari et al., 2021], but usually assume job values are externally given. Bandit methods have been used for online resource allocation in edge systems [Ouyang et al., 2021, Zhang et al., 2022, Han et al., 2021, Kim et al., 2024]. Classical stochastic bandits assume stationary rewards [Auer et al., 2002], while sliding-window, discounted, and change-point methods handle non-stationarity [Garivier and Moulines, 2008, Besbes et al., 2014, Besson et al., 2022]. Rotting and time-increasing bandits capture monotone reward evolution [Seznec et al., 2019, 2020, Xia et al., 2024]. Our setting differs because adapter gains saturate within a stable slot, but can be refreshed or invalidated by tenant replacement and drift under finite VRAM-slot constraints.

## 3 Preliminaries & Problem Formulation

### 3.1 System Abstraction: Slot-Based Decoupling

A multi-tenant edge adaptation service must decide both which tenant adapters reside in limited GPU memory and which resident adapter receives the next training update. We decouple these controls. A slow system layer maintains a GPU-resident active set under capacity constraints and tenant churn, while a fast learning layer schedules compute among the currently resident slots. This paper focuses on the learning-layer scheduler and treats the system-layer admission outcome as exogenous. The resulting slot abstraction gives the scheduler a fixed action space: each physical GPU slot is an arm, although the tenant occupying that slot may change over time due to eviction, repopulation, or distribution shift.

This decoupling is deliberate. Admission and eviction operate on a slower system timescale because they may involve CPU/SSD-GPU state movement, optimizer-state restoration, cache eviction, and service-priority constraints, whereas the learning layer only chooses among currently train-ready slots after each fine-tuning step. Jointly optimizing both layers would require reasoning over a large and time-varying tenant population with heterogeneous I/O costs.

### 3.2 Bandit Formulation

**Arms (GPU slots) and tenants.** A single edge node hosts a shared base model and can keep at most $K$ tenant-specific adapters resident in GPU memory at any time. We index the physical adapter slots by $\mathcal{A} = \{1, \dots, K\}$ and treat each slot as an *arm*. The (potentially unbounded) set of tenants is denoted by $\mathcal{U}$.

**System-layer admission as an exogenous environment.** At global time $t \in [T]$, each slot $i \in \mathcal{A}$ is occupied by the adapter of some tenant determined by the slow-timescale admission/eviction policy. We represent this outcome by an exogenous mapping $\psi_t : \mathcal{A} \to \mathcal{U}$, where $\psi_t(i)$ is the tenant whose adapter resides in slot $i$ at time $t$. Our bandit algorithm only decides *which slot to train in each round*; the mapping $\psi_t$ is independent of the scheduler's learning decisions. From the scheduler's viewpoint, changes in $\psi_t$ manifest as *external non-stationarity* of an arm (i.e., a slot's "identity" and reward behavior may change when it is repopulated).

**Action, observation, and net reward.** In each decision round $t$, the scheduler selects one arm $I_t \in \mathcal{A}$, corresponding to performing one incremental fine-tuning step on the adapter currently resident in slot $I_t$, using a mini-batch of samples drawn from tenant $\psi_t(I_t)$. This action produces an *incremental performance gain* $g_{I_t,t}$ for that tenant, such as a reduction in the tenant-level training loss, along with a lightweight monitoring signal $z_{I_t,t}$.

In our personalization setting, the mini-batch training loss has a specific interpretation: it measures how well the current tenant adapter fits the current tenant data stream. Therefore, it is a natural online signal of tenant-adapter mismatch and can be obtained without any extra forward passes beyond the training step itself. We instantiate $z_{I_t,t}$ using this training loss. Importantly, we do not treat it as a replacement for held-out validation perplexity or as a perfect measure of generalization. Instead, it is used as a noisy, low-overhead proxy for detecting persistent harmful shifts in the current tenant stream.

To reduce sensitivity to mini-batch composition and text difficulty, WCA-UCB never triggers a degradation reset from a single proxy value. The proxy is accumulated in a sliding window and compared only through windowed averages with a margin, as detailed in Section 4.2. Thus, the detector is designed to react to sustained changes rather than transient loss spikes.

We use a nonnegative per-step adaptation penalty $\lambda_{\text{cost}} \geq 0$ to model the marginal cost of an additional fine-tuning update, including GPU time, energy, thermal budget, and the opportunity cost of the training pipeline. The *net reward* is $r_{i,t} = g_{i,t} - \lambda_{\text{cost}}$. For positive adaptation cost, updates with marginal gain below this penalty have negative net reward; the zero-penalty configuration is also well-defined as an algorithmic setting. The goal is to maximize $\sum_{t=1}^{T} r_{I_t,t}$ over horizon $T$.

**Evolution of rewards (piecewise-convergence structure).** The reward distribution of each slot is non-stationary and evolves due to two coupled effects: (i) *convergence* of the tenant-specific adapter under repeated fine-tuning (an internal state change within a stable period), and (ii) *environment changes* such as tenant churn (context switches induced by changes in $\psi_t$) or distribution shifts in the tenant's data (external drift). We model these dynamics via a **piecewise-convergence structure** along the global time axis.

For each arm $i \in \mathcal{A}$, its evolution is decomposed into at most $M_i$ *segments* defined by change-points $1 = \tau_{i,1} < \tau_{i,2} < \cdots < \tau_{i,M_i} < \tau_{i,M_i+1} = T + 1$. Segment $j \in \{1, \dots, M_i\}$ of arm $i$ is the interval $S_{i,j} = [\tau_{i,j}, \tau_{i,j+1})$ on the global time axis. For brevity, we define the set of all such arm-segment pairs as $\mathcal{S} = \{(i, j) : i \in \mathcal{A}, 1 \leq j \leq M_i\}$, and refer to a specific segment as $(i, j) \in \mathcal{S}$. Importantly, the change-points $\{\tau_{i,j}\}$ are arm-specific and need not align across arms: at any time $t$, different arms may be in different segments, modeling asynchronous tenant activity and adapter convergence.

For each segment $(i, j) \in \mathcal{S}$, repeatedly fine-tuning the same adapter exhibits an *ascending-saturating* pattern: at the beginning of the segment, the expected net gain is positive, and it gradually decreases to zero (or below) as the adapter converges. To formalize this, we index the pulls of arm $i$ within the

interval $S_{i,j}$ by their local count $n = 1, 2, \ldots$. Specifically, for any time step $t \in S_{i,j}$ corresponding to the $n$-th pull of arm $i$, the reward is modeled as a random variable $r_{i,t}$ with conditional mean $\mu_{i,j}(n)$.

**Assumption 1** (Piecewise-convergence reward model)**.** *For each segment $(i, j) \in \mathcal{S}$, there exists an unknown* saturation time $\nu_{i,j} \in \mathbb{N}$ *and constants $\Delta_i^{\text{sat}} > 0$, $\tau_{\text{sat}} > 0$, and $B > 0$ such that: (1)* ***Ascending phase:*** *For all $1 \le n \le \nu_{i,j}$ the expected net reward is positive and uniformly lower bounded: $\mu_{i,j}(n) \ge \Delta_i^{\text{sat}}$, and the sequence $n \mapsto \mu_{i,j}(n)$ is non-increasing; (2)* ***Saturation phase:*** *For all $n \ge \nu_{i,j}$, the expected net reward drops significantly below zero to ensure detectability with $\mu_{i,j}(n) \le -2\tau_{\text{sat}}$ (Note: In practice, incremental gains often decay to near zero while the adaptation cost $\lambda_{\text{cost}}$ remains strictly positive, justifying this gap.); and (3)* ***Sub-Gaussian noise:*** *For all $i, j, n$, the centered rewards $r_{i,t} - \mu_{i,j}(n)$ are conditionally $B$-sub-Gaussian.*

**Assumption 2** (Proxy-detectable harmful shifts)**.** *For each arm segment $(i, j) \in \mathcal{S}$, the proxy signal $z_{i,t}$ has a segment-wise stationary mean $\theta_{i,j}$ and bounded noise. For unannounced harmful shifts that are not directly exposed by the system-layer slot mapping, there exists $\Delta_i^{\text{deg}} > 0$ such that $\theta_{i,j+1} \ge \theta_{i,j} + \Delta_i^{\text{deg}}$, i.e., the proxy becomes worse, such as a higher loss or entropy, by at least $\Delta_i^{\text{deg}}$ in expectation. The centered noise $z_{i,t} - \theta_{i,j}$ is $B_z$-sub-Gaussian.*

Assumption 1 models the ascending-saturating pattern of incremental adapter gains under a fixed tenant/environment. Assumption 2 is used only for *implicit harmful shifts*. Explicit tenant replacement does not require the proxy to become worse: when the system observes $\psi_t(i) \ne \phi_i$, WCA-UCB resets the slot identity directly. Benign or easier shifts may not trigger the one-sided degradation detector, but they also do not create the same urgent risk of exploiting stale high-reward histories under degraded performance.

**Remark 1.** *In practice, we may include a special "no-op" arm that corresponds to skipping adaptation in a round and simply using the current adapters and base model; its net reward is fixed at zero. In the analysis, this arm can be treated as an additional arm with a known mean of 0, and it does not affect the regret order.*

## 4 Methodology

We now present *W*indowed *C*onvergence-*A*ware *UCB* (WCA-UCB), which combines a sliding-window UCB estimator with gain-based saturation detection and proxy-based degradation-triggered resets. The main text describes the estimator and detection logic, while the complete pseudocode is provided in Appendix A. A schematic overview of the closed-loop execution is provided in Figure 2.

### 4.1 Sliding-window UCB with fixed-$\delta$ bonus

For robustness to non-stationarity within segments, we estimate the mean reward using only the most recent observations. Fix a reward-window size $W \in \mathbb{N}$. For arm $i$ and round $t$, let $\mathcal{R}_i(t) = \{s \le t : I_s = i, t - W < s \le t\}$ be the indices of the last $\min\{W, n_i(t)\}$ pulls of arm $i$, where $n_i(t) = |\mathcal{R}_i(t)|$ is the effective sample count. The windowed empirical mean reward is

$$\hat{\mu}_i^{(W)}(t) = (1/\max\{1, n_i(t)\}) \cdot \sum\nolimits_{s \in \mathcal{R}_i(t)} r_{i,s}. \tag{1}$$

Instead of a time-dependent bonus $\sqrt{\log t / n_i(t)}$, WCA-UCB uses a fixed-confidence style bonus controlled by a parameter $\delta_T \in (0, 1/2]$:

$$b_i(t) = c\sqrt{\Lambda_T / \max\{1, n_i(t)\}},\ \Lambda_T = \ln(1/\delta_T), \tag{2}$$

where $c > 0$ is a constant. In the theoretical analysis we choose $\delta_T = T^{-2}$ (or $\delta_T = (KT)^{-2}$), so that $\Lambda_T = O(\log T)$. In implementation, we instantiate $\delta_T$ as a small constant (e.g., 0.05), which empirically provides a good trade-off between optimism and stability.

The UCB index of arm $i$ at round $t$ is then $I_i(t) = \hat{\mu}_i^{(W)}(t) + b_i(t)$, with the convention that $I_i(t) = +\infty$ if $n_i(t) = 0$ to enforce initial exploration. A simple round-robin cold-start phase of length $n_0$ per arm, as implemented in our code, can be viewed as an explicit way to guarantee this initial exploration and only contributes an $O(Kn_0)$ additive term to the regret.

**Remark 2** (Windowed robustness)**.** *LoRA fine-tuning losses can fluctuate due to mini-batch composition and heterogeneous text difficulty. WCA-UCB therefore bases its UCB indices, saturation tests, and degradation tests on finite windows rather than single observations. This smooths transient spikes and makes resets depend on persistent changes.*

**Algorithmic overview.** At each round, WCA-UCB first checks whether the tenant occupying any slot has changed. If so, the corresponding slot history is reset because the past observations no longer describe the current adapter. For each eligible non-saturated slot, the scheduler computes a windowed UCB index using the recent reward window $\mathcal{R}_i$ and selects the slot with the largest index. After one incremental fine-tuning step, it observes the gain $g_{I_t,t}$ and proxy $z_{I_t,t}$, updates the reward, gain, and proxy windows, and then applies the saturation and degradation tests described below. The complete pseudocode is provided in Algorithm 1 in Appendix A.

### 4.2 Saturation and degradation detection

In addition to $\mathcal{R}_i(t)$, for each arm $i$ we maintain sliding windows $\mathcal{G}_i$ and $\mathcal{Z}_i$ of length at most $\omega_{\text{sat}}$ and $\omega_{\text{deg}}$ that store the most recent gains $g_{i,t}$ and proxy values $z_{i,t}$, respectively. Let $\bar{g}_i^{(\omega)}$ be the average of the values in $\mathcal{G}_i$ when the window is full, and let $\bar{z}_i^{\text{ref}}, \bar{z}_i^{\text{cur}}$ denote the averages over the older and more recent parts of $\mathcal{Z}_i$.

**Gain-based saturation detection.** When $|\mathcal{G}_i| = \omega_{\text{sat}}$, we compare $\bar{g}_i^{(\omega)}$ with $\lambda_{\text{cost}}$ using a margin $\tau_{\text{sat}} > 0$. If $\bar{g}_i^{(\omega)} \leq \lambda_{\text{cost}} - \tau_{\text{sat}}$, we interpret this as the adapter having reached saturation in its current segment: the marginal gain can no longer compensate the adaptation cost. WCA-UCB then triggers RESETARM$(i, \text{sat})$, which clears the sliding windows and marks arm $i$ as *saturated* so it will not be scheduled until the slot is refreshed, e.g., by a context switch. The full reset routine is given in Algorithm 2 in Appendix A. Optionally, RESETARM can notify the system layer to refresh slot $i$ via eviction or state reset, marking the end of the current segment.

**Proxy-based degradation detection.** The proxy detector is used for unannounced harmful shifts, such as data-distribution changes within the same tenant, where the system layer may not provide an explicit event. When $|\mathcal{Z}_i| = \omega_{\text{deg}}$, we split $\mathcal{Z}_i$ into an older part $\mathcal{Z}_i^{\text{ref}}$ and a recent part $\mathcal{Z}_i^{\text{cur}}$. If $\bar{z}_i^{\text{cur}} - \bar{z}_i^{\text{ref}} \geq \tau_{\text{deg}}$, the recent proxy has become persistently worse and WCA-UCB triggers RESETARM$(i, \text{deg})$, which discards stale history while keeping the arm eligible for re-adaptation. In contrast, explicit tenant replacement is handled by the slot-identity check $\psi_t(i) \neq \phi_i$ in Algorithm 1; it does not require the new tenant's initial loss to be higher.

**Reset operation.** Both saturation and degradation events call the same reset primitive. The reset clears the selected slot's effective sample count and sliding-window histories $(\mathcal{R}_i, \mathcal{G}_i, \mathcal{Z}_i)$. For a saturation reset, the slot is marked ineligible until the slot is refreshed; for a degradation reset, the slot remains eligible so that WCA-UCB can immediately re-explore the new regime. The detailed reset procedure is provided in Algorithm 2 in Appendix A.

## 5 Theoretical Analysis

In this section, we establish the theoretical guarantees of WCA-UCB. We analyze the pseudo-regret of WCA-UCB against a dynamic oracle that, at each time $t$, selects the arm with the highest ascending-phase reward in its current segment. Since segments are asynchronous across arms, we explicitly define the segment index $J_i(t)$ such that $t \in S_{i,J_i(t)}$ for each arm $i$. The oracle's reference mean reward at time $t$ is naturally given by: $\mu_t^* = \max_{k \in \mathcal{A}} \bar{\mu}_{k,J_k(t)}$.

We now characterize the hardness of the learning problem via the minimum reward gap against this oracle. The following assumption ensures that the optimal arm is distinguishable from sub-optimal ones.

**Assumption 3** (Segment-wise gaps against dynamic oracle)**.** *For each segment $(i,j) \in \mathcal{S}$, let $\bar{\mu}_{i,j}$ be its average net reward in the ascending phase. We define the gap of arm $i$ in segment $j$ as $\Delta_{i,j} = \inf_{t:J_i(t)=j}(\mu_t^* - \bar{\mu}_{i,j})$, where the infimum is taken over times $t$ where arm $i$ is in segment $j$ and still ascending. We assume $\Delta_{i,j} > 0$ for all sub-optimal pairs $(i,j) \in \mathcal{S}$ and denote $\Delta_{\min} = \min_{(i,j)\in\mathcal{S}:\Delta_{i,j}>0} \Delta_{i,j}$.*

Based on the oracle defined above, the cumulative regret of Algorithm 1 is defined as: $R(T) = \sum_{t=1}^{T}(\mu_t^* - \mathbb{E}[r_{I_t,t}])$.

Our main result shows that under Assumptions 1-3, WCA-UCB with a fixed-$\delta$ bonus, saturation detection and reset achieves a regret bound of the same logarithmic order as classical UCB, up to a multiplicative factor that depends on the total number of segments and an additive term that scales linearly in the detection window size.

**Theorem 1** (Regret of WCA-UCB with saturation and reset)**.** *Suppose Assumptions 1-3 hold and choose $\tau_{\text{sat}}, \tau_{\text{deg}} > 0$, detection window sizes $\omega_{\text{sat}}, \omega_{\text{deg}} \in \mathbb{N}$, reward window size $W \in \mathbb{N}$, and confidence parameter $\delta_T \in (0, 1/2]$ such that:*

*(i) (**Correct saturation detection**) For any arm $i$ and segment $j$, as long as the arm is still in the ascending phase ($n \le \nu_{i,j}$), the gain window average $\bar{g}_i^{(\omega)}$ satisfies $\mathbb{E}[\bar{g}_i^{(\omega)}] \ge \lambda_{\text{cost}} + 2\tau_{\text{sat}}$. Once it enters the saturation phase ($n \ge \nu_{i,j}$), we have $\mathbb{E}[\bar{g}_i^{(\omega)}] \le \lambda_{\text{cost}} - 2\tau_{\text{sat}}$.*

*(ii) (**Detectable degradation**) For any arm $i$ and consecutive segments $j$ and $j+1$, the proxy means satisfy $\theta_{i,j+1} - \theta_{i,j} \ge 4\tau_{\text{deg}}$.*

*Let $M_i$ be the number of segments of arm $i$ and $M = \sum_{i=1}^{K} M_i$. Let $\Lambda_T = \ln(1/\delta_T)$ be the confidence parameter appearing in the bonus (2). Then there exist constants $C_1, C_2 > 0$ depending only on $(B, B_z, c, \Delta_i^{\text{sat}}, \Delta_i^{\text{deg}}, W)$ such that*

$$R(T) \le C_1 \Lambda_T \sum_{(i,j)\in\mathcal{S}} 1/\Delta_{i,j}^2 + C_2\, M\, \max\{\omega_{\text{sat}}, \omega_{\text{deg}}\} + O(1). \tag{3}$$

*The first term is the segment-wise exploration cost, while the second term is the window-scale tracking cost incurred by saturation and degradation detection after regime changes. In particular, if $M = O(1)$ and $W, \omega_{\text{sat}}, \omega_{\text{deg}} = O(\log T)$, and we set $\delta_T = T^{-2}$ so that $\Lambda_T = O(\log T)$, then*

$$R(T) = \tilde{O}\left(\log T/\Delta_{\min}^2\right), \tag{4}$$

*i.e., WCA-UCB achieves a logarithmic regret bound of the same order as convergence-aware UCB in the single-segment setting, even when segments are asynchronous across arms.*

**Scope of the zero-cost setting.** WCA-UCB is well-defined when $\lambda_{\text{cost}} = 0$, but the above guarantee remains conditional on the stated reward-separation assumptions. In particular, at zero adaptation cost, the post-saturation negative net-reward gap requires the expected marginal gain itself to become sufficiently negative. If the marginal gain only approaches zero from above, the present saturation-detection guarantee does not apply. We make no general zero-cost regret claim beyond these conditions.

**Proof outline.** The complete proof is deferred to Appendix C. At a high level, the analysis combines windowed concentration bounds, segment-wise UCB regret, and bounded delay costs caused by saturation and degradation detection. This yields the first term in (3) from exploration within stable ascending phases, and the second term from tracking regime changes and detector delays.

## 6 Performance Evaluation

### 6.1 Experimental Setup

We evaluate WCA-UCB using two complementary workloads: trace-driven simulations for controlled regret analysis and a multi-tenant LoRA fine-tuning prototype for model-quality evaluation. For the trace-driven simulations, we generate synthetic traces following the ascending-saturating model in Assumption 1, inject Gaussian observation noise, and add random arm-specific change-points to emulate asynchronous tenant churn and distribution shifts. For the prototype workload, we use UltraChat, a large-scale instruction-following dialogue corpus, to construct $N = 20$ shadow tenants with heterogeneous task distributions, including code, math, and creative writing. The prototype runs Qwen2.5-1.5B-Instruct with BF16 LoRA adapters ($r = 8, \alpha = 16$) on a constrained single-GPU setup with $K = 5$ train-ready slots, and multiplexes these tenants into slot sessions to induce tenant

Table 1: Multi-tenant LoRA prototype results.

| Algorithm | PPL ↓ | Proxy Loss ↓ | Target Switches ↓ |
|---|---|---|---|
| **WCA_UCB (Ours)** | **3.8456** | **1.3469** | 278 |
| GLR_klUCB | 3.9504 | 1.3738 | 1609 |
| D_UCB | 3.9767 | 1.3805 | 351 |
| SW_UCB | 4.2082 | 1.4370 | 252 |
| Greedy | 4.1058 | 1.4124 | 658 |
| Random | 4.0219 | 1.3918 | 1994 |
| RoundRobin | 4.0261 | 1.3928 | 2500 |
| RAW_UCB | 4.6005 | 1.5262 | 23 |

churn and distribution shifts. The mini-batch training loss is used as a noisy low-overhead proxy for windowed shift detection, not as a replacement for offline PPL-based model-quality evaluation.

**Baselines.** We compare WCA-UCB with Random, Round-Robin, Greedy with $\epsilon$-greedy exploration, RAW-UCB [Seznec et al., 2020], SW-UCB/D-UCB [Garivier and Moulines, 2008], GLR-klUCB [Besson et al., 2022], and an Oracle used only for regret calculation.

**Metrics.** We report cumulative dynamic regret, cumulative average PPL, geometric-mean PPL, average proxy loss, relative loss gap, and training-target switches. Cumulative average PPL tracks the long-term model-quality trend, while geometric-mean PPL summarizes prototype performance across tenant evaluations. Average proxy loss reports the mini-batch training loss used for online monitoring. Training-target switches count changes of the selected resident slot between consecutive update rounds, i.e., $I_t \neq I_{t-1}$; they measure scheduling stability among resident adapters rather than physical CPU/SSD-GPU adapter swapping or direct I/O latency.

**Reward-structure validation.** Before comparing schedulers, we run a controlled three-phase LoRA adaptation experiment to verify the qualitative reward structure. As shown in Figure 1, validation PPL first drops and plateaus under a stable code-oriented tenant stream, showing diminishing marginal gains. An intra-tenant shift to math data and a later slot repopulation by a general-chat tenant both increase PPL, indicating that stale histories become unreliable after harmful shifts. This supports the piecewise-converging abstraction and motivates gain-based saturation detection plus proxy-based reset.

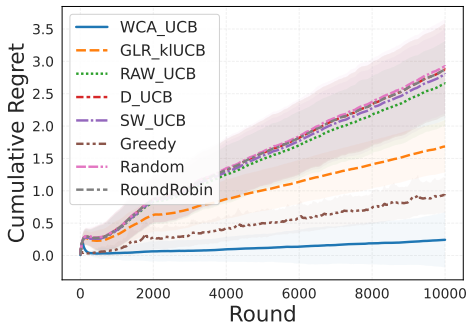
### 6.2 Main Results

We next compare WCA-UCB with bandit and heuristic baselines in terms of regret, model quality, and switching behavior. Trace-driven simulations are averaged over 1000 random seeds and evaluated against an Oracle with ground-truth segment information.

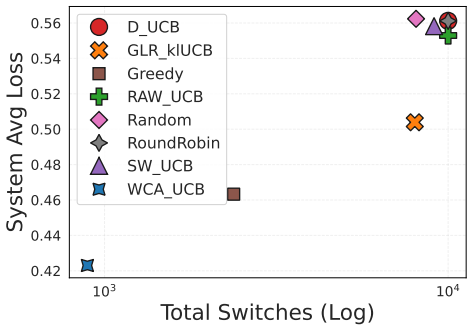
Figure 3 summarizes the trace-driven simulation results. WCA-UCB achieves the lowest cumulative regret and avoids the linear growth of passive non-stationary baselines such as SW-UCB and D-UCB, which forget old observations but do not explicitly handle convergence saturation. This regret advantage translates into lower cumulative average PPL and a consistently positive relative loss gap over all competitors. In the Pareto plot, WCA-UCB attains the best quality-switching trade-off: it avoids the high-switching regime of exploratory baselines while achieving substantially lower loss than conservative policies such as Greedy.

**Multi-tenant LoRA prototype.** On the prototype workload constructed from UltraChat, WCA-UCB achieves the best geometric-mean PPL and proxy loss while using far fewer training-target switches than GLR-klUCB. As shown in Table 1, WCA-UCB improves PPL over GLR-klUCB (3.8456 vs. 3.9504) while reducing training-target switches from 1609 to 278, indicating a more stable update allocation among resident slots. RAW-UCB switches rarely but has the worst PPL, indicating insufficient adaptation after distribution shifts.

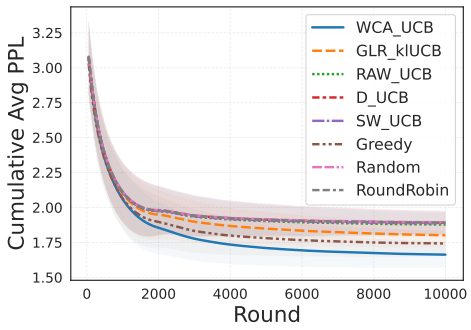
Figure 4 further shows the same quality-switching trend over time: WCA-UCB reaches a lower cumulative average PPL while avoiding the excessive switching of highly exploratory baselines.



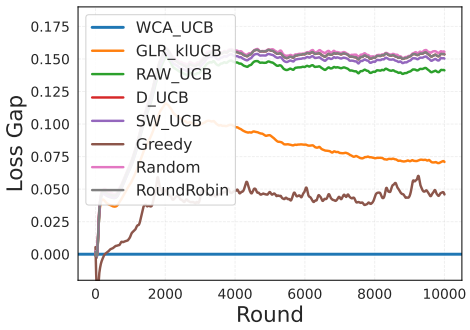

(a) Cumulative Regret.




(b) Pareto Efficiency.

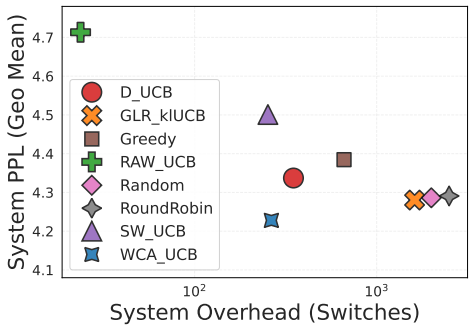



(c) Cumulative Avg PPL.

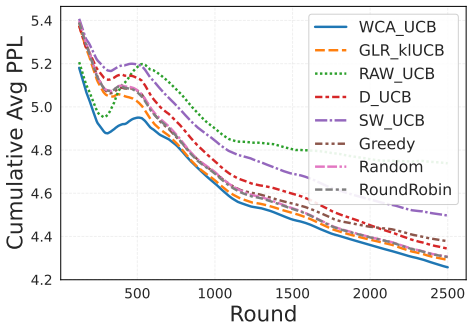



(d) Relative Loss Gap.

Figure 3: Trace-driven simulation results. (a) Cumulative dynamic regret. (b) Average loss versus training-target switches. (c) Cumulative average PPL. (d) Relative loss gap against WCA-UCB.




(a) Pareto Efficiency.




(b) Cumulative Avg PPL.

Figure 4: Prototype evaluation on the UltraChat-derived multi-tenant workload. (a) Geometric-mean PPL versus training-target switches. (b) Cumulative average PPL over 2500 fine-tuning rounds.

### 6.3 Ablation and Sensitivity

Additional ablation and sensitivity results are reported in Appendix B. The ablation curves show that removing drift detection causes linear regret growth and stagnant PPL, while removing the sliding window slows convergence and yields worse final PPL. Parameter sensitivity results further show that WCA-UCB remains stable across broad choices of $W$, $\eta$, $\Omega$, and $\lambda_{\text{cost}}$, indicating that its gain does not depend on a narrowly tuned configuration.

**Robustness of the degradation detector.** A focused 100-seed diagnostic tests isolated loss spikes, finite hard-example bursts, and a persistent proxy-mean shift (Appendix B, Table 2). The detector triggers no resets for isolated one-round spikes and resets in 15% of eight-round hard bursts, versus 95% for a persistent shift. However, 32-round hard bursts trigger resets in 85% of trials, indicating that loss-window monitoring cannot reliably distinguish sufficiently long difficult-sample bursts from sustained degradation.

## 7 Limitations

Our analysis relies on piecewise-converging rewards, detectable post-saturation reward separation, segment-wise stable proxy means, sufficiently persistent harmful shifts, and the specified noise conditions. In practice, prolonged hard-example bursts may cause false resets, while shifts below the detector threshold may be delayed or missed; loss alone does not identify the semantic cause of a change. The model uses a homogeneous per-update penalty rather than state-dependent transfer costs, and training-target switches count logical changes in the selected resident slot, not measured HBM/DRAM/NVMe transfers or I/O latency. Finally, the Qwen2.5-1.5B prototype with five train-ready slots on a single RTX 4090 is a proof of concept, not evidence of effectiveness on larger backbones, production edge accelerators, or larger resident-slot configurations.

## 8 Conclusion

We studied multi-tenant incremental LoRA adaptation under a strict GPU-resident slot budget. The key difficulty is that marginal update gains decay as an adapter saturates, while slot repopulation and distribution shifts can invalidate stale histories. We proposed WCA-UCB, a windowed UCB scheduler with gain-based saturation detection and proxy-based reset. Under the stated conditions, the analysis gives a dynamic regret bound separating within-segment exploration cost from regime-change tracking cost, with a logarithmic form under bounded-change conditions. Trace-driven simulations and an edge-like single-GPU LoRA prototype show that WCA-UCB improves perplexity and reduces training-target switches relative to the evaluated non-stationary bandit baselines. Future work includes evaluating simple early-stopping schedulers, measuring actual adapter state-transfer overhead on edge accelerators, and jointly optimizing admission and update scheduling.

## Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. U25A20436 and No. 62202016, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2025A1515012843, the Start-up Fund from Beijing Normal University under Grant No. 312200502510, the Internal Fund from Beijing Normal-Hong Kong Baptist University under Grant No. UICR0400003-24 and No. UICR0200022-25, and the Interdisciplinary Intelligence SuperComputer Center of Beijing Normal University (Zhuhai).

## References

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Firas Bayram, Bestoun S Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245:108632, 2022.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

Lilian Besson, Emilie Kaufmann, Odalric-Ambrym Maillard, and Julien Seznec. Efficient change-point detection for tackling piecewise-stationary bandits. *Journal of Machine Learning Research*, 23(77):1–40, 2022.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant lora serving. *Proceedings of Machine Learning and Systems*, 6:1–13, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Tobias Domhan, Jost Tobias Springenberg, Frank Hutter, et al. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, volume 15, pages 3460–8, 2015.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys*, 46(4):1–37, 2014.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

Juncheng Gu, Mosharaf Chowdhury, Kang G Shin, Yibo Zhu, Myeongjae Jeon, Junjie Qian, Hongqiang Liu, and Chuanxiong Guo. Tiresias: A {GPU} cluster manager for distributed deep learning. In *16th USENIX Symposium on Networked Systems Design and Implementation*, pages 485–500, 2019.

Yuqi Han, Lihua Ai, Rui Wang, Jun Wu, Dian Liu, and Haoqi Ren. Cache placement optimization in mobile edge computing networks with unaware environment—an extended multi-armed bandit approach. *IEEE Transactions on Wireless Communications*, 20(12):8119–8133, 2021.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Taejin Kim, Jinhang Zuo, Xiaoxi Zhang, and Carlee Joe-Wong. Edge-msl: Split learning on the mobile edge via multi-armed bandits. In *IEEE Conference on Computer Communications*, pages 391–400. IEEE, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4582–4597, 2021.

Tao Ouyang, Xu Chen, Zhi Zhou, Rui Li, and Xin Tang. Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning. *IEEE Transactions on Mobile Computing*, 22(3):1313–1326, 2021.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14, 2021.

Julien Seznec, Andrea Locatelli, Alexandra Carpentier, Alessandro Lazaric, and Michal Valko. Rotting bandits are no harder than stochastic ones. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2564–2572. PMLR, 2019.

Julien Seznec, Pierre Menard, Alessandro Lazaric, and Michal Valko. A single algorithm for both restless and rested rotting bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 3784–3794. PMLR, 2020.

Zheyu Shen, Yexiao He, Ziyao Wang, Yuning Zhang, Guoheng Sun, Wanghao Ye, and Ang Li. Edgelora: An efficient multi-tenant llm serving system on edge devices. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*, pages 138–153, 2025.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*, 2023.

Chunlin Tian, Xinpeng Qin, Kahou Tam, Li Li, Zijian Wang, Yuanzhe Zhao, Minglei Zhang, and Chengzhong Xu. {CLONE}: Customizing {LLMs} for efficient {Latency-Aware} inference at the edge. In *2025 USENIX Annual Technical Conference*, pages 563–585, 2025.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. {dLoRA}: Dynamically orchestrating requests and adapters for {LoRA}{LLM} serving. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pages 911–927, 2024.

Yu Xia, Fang Kong, Tong Yu, Liya Guo, Ryan A Rossi, Sungchul Kim, and Shuai Li. Which llm to play? convergence-aware online model selection with time-increasing bandits. In *Proceedings of the ACM Web Conference 2024*, pages 4059–4070, 2024.

Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, et al. Gandiva: Introspective cluster scheduling for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation*, pages 595–610, 2018.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024.

Peifeng Yu and Mosharaf Chowdhury. Fine-grained gpu sharing primitives for deep learning applications. *Proceedings of Machine Learning and Systems*, 2:98–111, 2020.

Rui Zhang, Peng Cheng, Zhuo Chen, Sige Liu, Branka Vucetic, and Yonghui Li. Calibrated bandit learning for decentralized task offloading in ultra-dense networks. *IEEE Transactions on Communications*, 70(4):2547–2560, 2022.

# A Detailed Pseudocode

This section provides the complete pseudocode of WCA-UCB and its reset primitive. The main text describes the estimator, saturation detector, and proxy-based degradation detector; Algorithm 1 gives the full per-round execution flow.

**Algorithm 1:** WCA-UCB: Windowed UCB with Saturation/Degradation Reset

**Input:** $K, T, W, \omega_{\text{sat}}, \omega_{\text{deg}}, c, \delta_T, \lambda_{\text{cost}}, \tau_{\text{sat}}, \tau_{\text{deg}}$

**for** $i \leftarrow 1$ **to** $K$ **do**
  $n_i \leftarrow 0$; $(\mathcal{R}_i, \mathcal{G}_i, \mathcal{Z}_i) \leftarrow \emptyset$; $s_i \leftarrow 0$; $\phi_i \leftarrow \perp$
**for** $t \leftarrow 1$ **to** $T$ **do**
  **for** $i \leftarrow 1$ **to** $K$ **do**
    **if** $\psi_t(i) \neq \phi_i$ **then**
      $\phi_i \leftarrow \psi_t(i)$
      RESETARM$(i, \text{deg})$ `// context switch`
  **for** $i \leftarrow 1$ **to** $K$ **do**
    **if** $s_i = 1$ **then**
      $I_i(t) \leftarrow -\infty$ `// saturated`
    **else**
      **if** $n_i = 0$ **then**
        $I_i(t) \leftarrow +\infty$ `// cold start`
      **else**
        $\hat{\mu}_i^{(W)}(t) \leftarrow \text{mean}(\mathcal{R}_i)$
        $I_i(t) \leftarrow \hat{\mu}_i^{(W)}(t) + c\sqrt{\ln(1/\delta_T)/n_i}$
  $I_t \in \arg\max_i I_i(t)$
  Fine-tune one step on slot $I_t$ (tenant $\psi_t(I_t)$)
  Observe $g_{I_t,t}$ and $z_{I_t,t}$
  $r_{I_t,t} \leftarrow g_{I_t,t} - \lambda_{\text{cost}}$
  Append $r_{I_t,t}$ to $\mathcal{R}_{I_t}$ (keep last $W$)
  $n_{I_t} \leftarrow \min\{W, n_{I_t} + 1\}$
  Append $g_{I_t,t}$ to $\mathcal{G}_{I_t}$ (keep last $\omega_{\text{sat}}$) and $z_{I_t,t}$ to $\mathcal{Z}_{I_t}$ (keep last $\omega_{\text{deg}}$)
  **if** $|\mathcal{G}_{I_t}| = \omega_{\text{sat}}$ ***and*** $\text{mean}(\mathcal{G}_{I_t}) \leq \lambda_{\text{cost}} - \tau_{\text{sat}}$ **then**
    RESETARM$(I_t, \text{sat})$ `// saturation`
  **if** $|\mathcal{Z}_{I_t}| = \omega_{\text{deg}}$ **then**
    $\bar{z}_{I_t}^{\text{ref}} \leftarrow \text{mean}(\text{first } 3\omega_{\text{deg}}/4 \text{ of } \mathcal{Z}_{I_t})$
    $\bar{z}_{I_t}^{\text{cur}} \leftarrow \text{mean}(\text{last } \omega_{\text{deg}}/4 \text{ of } \mathcal{Z}_{I_t})$
    **if** $\bar{z}_{I_t}^{\text{cur}} - \bar{z}_{I_t}^{\text{ref}} \geq \tau_{\text{deg}}$ **then**
      RESETARM$(I_t, \text{deg})$ `// degradation`

**Algorithm 2:** RESETARM$(i, \kappa)$

**Input:** Arm index $i$, reset type $\kappa \in \{\text{sat}, \text{deg}\}$

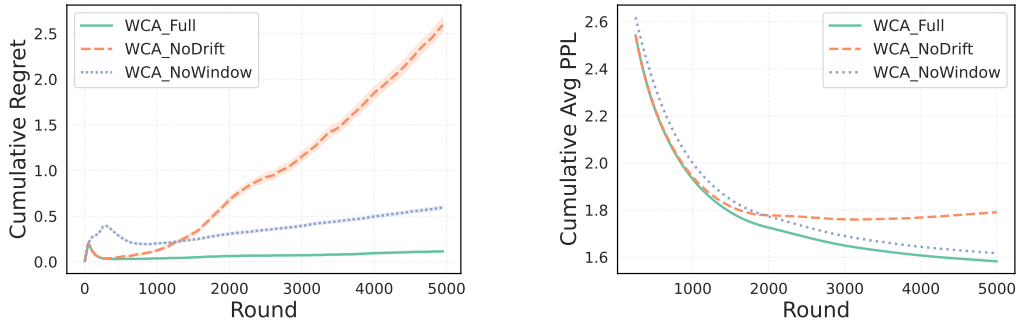
$n_i \leftarrow 0$; $(\mathcal{R}_i, \mathcal{G}_i, \mathcal{Z}_i) \leftarrow \emptyset$
**if** $\kappa = \text{sat}$ **then**
  $s_i \leftarrow 1$ `// mark saturated, ineligible until slot refresh`
**else**
  $s_i \leftarrow 0$ `// keep eligible, re-adapt under new regime`
`// optional: notify system layer to refresh slot` $i$ `via eviction or state reset`

**Remark 3** (Implementation Details)**.** *To enforce the initial exploration ($I_i(t) = +\infty$ when $n_i(t) = 0$), we implement a short round-robin cold-start phase where each arm is pulled $n_0$ times before the UCB-based selection begins. This guarantees a minimum number of samples for each arm and only contributes a negligible additive $O(Kn_0)$ term to the theoretical regret. Furthermore, one may maintain exponentially weighted moving averages (EWMA) of $g_{i,t}$ or $z_{i,t}$ as auxiliary statistics for added stability; however, our theoretical analysis relies strictly on the finite-window sample means.*

## B Additional Ablation and Parameter Sensitivity Results

This section provides the ablation and parameter sensitivity results omitted from the main text.

**Ablation study.** Figure 5 compares the full WCA-UCB algorithm with two variants: *NoDrift*, which disables proxy-based degradation detection, and *NoWindow*, which removes sliding-window reward estimation. Removing drift detection causes linear regret growth and stagnant PPL, showing that reset is necessary for adapting to changing tenants. Removing the sliding window also degrades performance, confirming that finite-window estimates improve robustness under noisy rewards.




(a) Cumulative Regret. (b) Cumulative Avg PPL.

Figure 5: Ablation study comparing the full WCA-UCB algorithm against variants without drift detection (*NoDrift*) and without a sliding window (*NoWindow*).

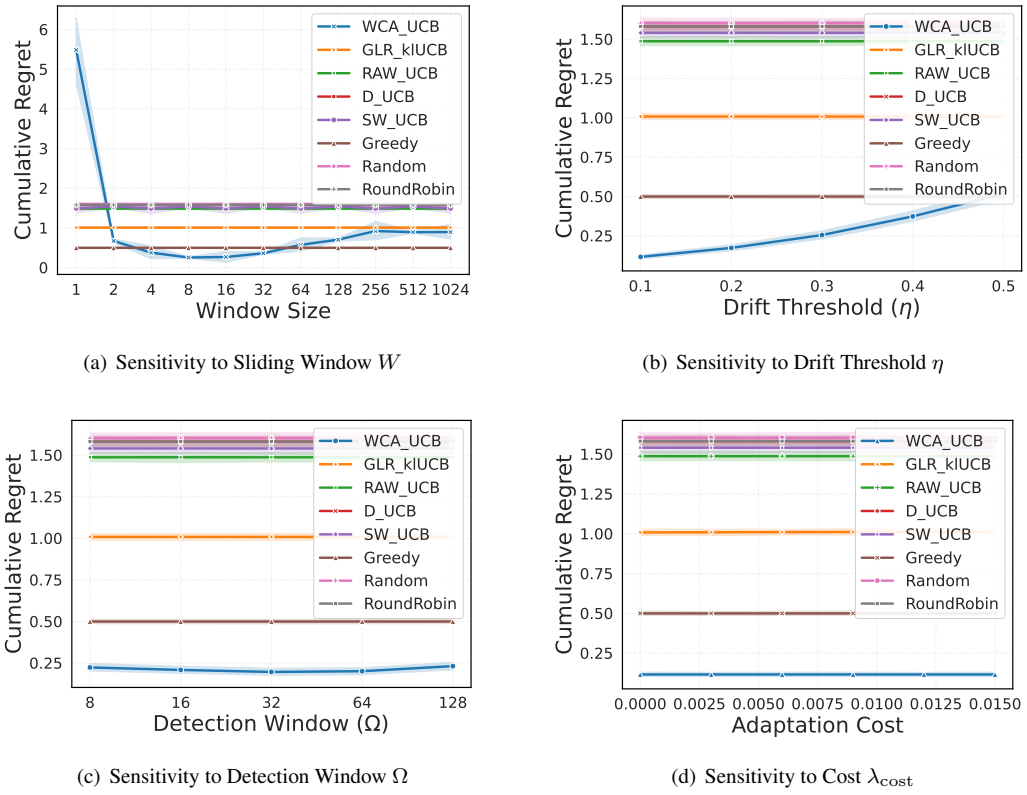
**Parameter sensitivity.** We vary the sliding window size $W$, drift threshold $\eta$ (corresponding to $\tau_{\text{deg}}$), detection window size $\Omega$ (which sets both $\omega_{\text{sat}}$ and $\omega_{\text{deg}}$), and adaptation cost $\lambda_{\text{cost}}$. Unless otherwise specified, the default configuration is $W = 16$, $\eta = 0.3$, $\Omega = 32$, and $\lambda_{\text{cost}} = 0.0$. The reported metric is the final cumulative regret. The $\lambda_{\text{cost}} = 0$ sensitivity result is empirical and should not be interpreted as a general zero-cost regret guarantee when the theorem's reward-separation conditions do not hold.

As shown in Figure 6, WCA-UCB consistently achieves low regret across a wide range of parameter configurations. The algorithm performs robustly for window sizes $W \in [16, 128]$, with only mild degradation when $W$ is extremely small. Increasing the drift threshold $\eta$ makes the detector less sensitive, but WCA-UCB maintains a clear advantage over the baselines. The method is also insensitive to the detection window size $\Omega$ over $\Omega \in [8, 128]$, and remains robust under different choices of the adaptation cost $\lambda_{\text{cost}}$. These results indicate that WCA-UCB does not rely on a single brittle hyperparameter configuration.

**Stress test of the proxy degradation detector.** In a 100-seed diagnostic, we separately consider an isolated one-round loss spike, bursts of difficult samples lasting eight or 32 rounds, and a persistent proxy-mean shift. Table 2 reports reset probability and detection delay using the stated threshold of 0.3. A sufficiently long hard-example burst can trigger a reset without a semantic distribution shift; the proxy is therefore a low-overhead warning signal rather than a reliable identifier of the cause of degradation.

## C Technical Proofs - Detailed Regret Analysis of WCA-UCB

In this section we provide a detailed proof of Theorem 1. For clarity, we first restate the dynamic oracle and regret notion, then construct high-probability concentration events, and finally bound the regret contributions from ascending phases, saturation phases, and degradation periods.




(a) Sensitivity to Sliding Window $W$

(b) Sensitivity to Drift Threshold $\eta$

(c) Sensitivity to Detection Window $\Omega$

(d) Sensitivity to Cost $\lambda_{\text{cost}}$

Figure 6: Parameter sensitivity analysis. Impact of (a) sliding window size $W$, (b) drift threshold $\eta$, (c) detection window $\Omega$, and (d) adaptation cost $\lambda_{\text{cost}}$ on the final cumulative regret.

Table 2: Proxy degradation-detector stress test (100 seeds, threshold 0.3), Dashes indicate that detection delays were not reported for trials without resets.

| Scenario | Reset probability | Median delay | P90 delay |
|---|---|---|---|
| Isolated 1-round spike | 0% | — | — |
| 8-round hard burst | 15% | 7 | 8 |
| 32-round hard burst | 85% | 7 | 8 |
| Persistent shift | 95% | 7 | 8 |

### C.1 Preliminaries and Dynamic Oracle

Recall that for each arm $i$ and time $t$, $J_i(t)$ denotes the index $j$ such that $t \in S_{i,j}$, where $\{S_{i,j}\}_{j=1}^{M_i}$ are the segments of arm $i$ along the global time axis. Each arm may have its own segment boundaries, so the indices $J_i(t)$ are generally *asynchronous* across arms.

Within segment $j$ of arm $i$, the expected net reward at the $n$-th pull in that segment is $\mu_{i,j}(n)$, and we define the average net reward over the ascending phase as

$$\bar{\mu}_{i,j} \;=\; \frac{1}{\nu_{i,j}} \sum_{n=1}^{\nu_{i,j}} \mu_{i,j}(n).$$

At global time $t$, when arm $i$ is in segment $J_i(t)$ and still in its ascending phase (i.e., its local pull count in that segment satisfies $n \le \nu_{i,J_i(t)}$), we compare to an oracle that always plays the arm with the highest ascending-phase average *among all arms at their current segments*. Formally, we define

$$\mu_t^* \;=\; \max_{i \in \mathcal{A}} \bar{\mu}_{i,J_i(t)}. \tag{5}$$

Note that this oracle benchmark effectively assumes the capability to always serve tenants in their ascending phases (e.g., by immediately switching to a new optimal tenant once saturation occurs), providing a strict upper bound on achievable performance. The dynamic pseudo-regret of WCA-UCB over horizon $T$ is

$$R(T) \;=\; \sum_{t=1}^{T} \Big( \mu_t^* - \mathbb{E}\big[r_{I_t,t}\big] \Big). \tag{6}$$

### C.2 High-Probability Concentration Events

We next construct three high-probability events that jointly control the accuracy of the windowed reward estimates, saturation detection, and degradation detection. Throughout we reuse the notation $\Lambda_T = \ln(1/\delta_T)$, with $\delta_T \in (0, 1/2]$ to be specified.

**Windowed reward concentration** Fix an arm $i$ and a segment $j$. Consider a time $t$ such that: (i) $J_i(t) = j$ (arm $i$ is in segment $j$), and (ii) all pulls of arm $i$ that fall into the reward window $\mathcal{R}_i(t)$ occur within the ascending phase of segment $j$, i.e., their local pull counts satisfy $n \le \nu_{i,j}$. Conditioned on this event, each reward in $\mathcal{R}_i(t)$ can be written as

$$r_{i,s} \;=\; \mu_{i,j}(n_s) + \xi_{i,s},$$

where $\xi_{i,s}$ is a $B$-sub-Gaussian noise term and $n_s$ is the local pull count for that sample. Define the deviation from the ascending-phase average as

$$\hat{\mu}_i^{(W)}(t) - \bar{\mu}_{i,j} \;=\; \frac{1}{n_i(t)} \sum_{s \in \mathcal{R}_i(t)} \big( r_{i,s} - \bar{\mu}_{i,j} \big),$$

where $n_i(t) = |\mathcal{R}_i(t)|$.

We decompose this deviation into two parts:

$$\hat{\mu}_i^{(W)}(t) - \bar{\mu}_{i,j} \;=\; \underbrace{\frac{1}{n_i(t)} \sum_{s \in \mathcal{R}_i(t)} \xi_{i,s}}_{\text{(zero-mean noise)}} \;+\; \underbrace{\frac{1}{n_i(t)} \sum_{s \in \mathcal{R}_i(t)} \big( \mu_{i,j}(n_s) - \bar{\mu}_{i,j} \big)}_{\text{(bias due to monotone means)}}.$$

By Assumption 1(c) and sub-Gaussian tails, the first term can be controlled via Hoeffding's inequality. The second term is a deterministic bias bounded in magnitude by a constant depending on $\sup_{n \le \nu_{i,j}} |\mu_{i,j}(n) - \bar{\mu}_{i,j}|$, which can be absorbed into the constants in our regret bound. Formally, we have the following lemma.

**Lemma 1** (Windowed reward concentration)**.** *Fix arm $i$ and segment $j$. There exists a constant $C_{\mathrm{bias}} > 0$ (depending on the variation of $\mu_{i,j}(n)$ in the ascending phase) such that for any time $t$ with $J_i(t) = j$ and all samples in $\mathcal{R}_i(t)$ coming from the ascending phase of segment $j$, we have*

$$\mathbb{P}\Big( \big| \hat{\mu}_i^{(W)}(t) - \bar{\mu}_{i,j} \big| \;>\; b_i(t) + C_{\mathrm{bias}} \Big) \;\le\; 2 \exp\!\Big( - \frac{n_i(t)\, b_i(t)^2}{2B^2} \Big),$$

*where $b_i(t)$ is the fixed-$\delta$ bonus defined in (2).*

*Proof.* Condition on $J_i(t) = j$ and on the event that all samples in $\mathcal{R}_i(t)$ come from the ascending phase. The noise terms $\{\xi_{i,s}\}$ are conditionally independent and $B$-sub-Gaussian, so Hoeffding's inequality yields

$$\mathbb{P}\Big( \Big| \frac{1}{n_i(t)} \sum_{s \in \mathcal{R}_i(t)} \xi_{i,s} \Big| > b_i(t) \Big) \;\le\; 2 \exp\!\Big( - \frac{n_i(t)\, b_i(t)^2}{2B^2} \Big).$$

The bias term is deterministic and bounded by

$$\Big| \frac{1}{n_i(t)} \sum_{s \in \mathcal{R}_i(t)} \big( \mu_{i,j}(n_s) - \bar{\mu}_{i,j} \big) \Big| \;\le\; \sup_{n \le \nu_{i,j}} |\mu_{i,j}(n) - \bar{\mu}_{i,j}| \;=:\; C_{\mathrm{bias}}.$$

Combining the two pieces via the triangle inequality proves the claim. □

By choosing $\delta_T$ such that $b_i(t)$ is on the order of $\sqrt{\Lambda_T / n_i(t)}$ with $\Lambda_T = \ln(1/\delta_T)$, and applying a union bound over all $(i, j)$ and all relevant time indices $t$, we can ensure that with high probability all windowed means stay within their confidence intervals around $\bar{\mu}_{i,j}$ up to an additive constant. We formalize this as follows.

**Lemma 2** (Global reward concentration event)**.** *Let $\delta_T \in (0, 1/2]$ and $\Lambda_T = \ln(1/\delta_T)$. There exists an event $\mathcal{E}_\mu$ with $\mathbb{P}(\mathcal{E}_\mu) \geq 1 - O(\delta_T)$ such that on $\mathcal{E}_\mu$, for all arms $i$, all segments $j$, and all times $t$ with $J_i(t) = j$ and window samples in the ascending phase,*

$$\left|\hat{\mu}_i^{(W)}(t) - \bar{\mu}_{i,j}\right| \;\leq\; b_i(t) + C_{\text{bias}}.$$

*Proof.* Apply Lemma 1 with a suitably chosen $b_i(t)$ such that the right-hand side probability is at most $\delta_T / (K \max_i M_i T)$, and take a union bound over all arms, segments, and relevant time steps. □

**Saturation and degradation detection** We now control the accuracy and delay of the gain-based saturation detector and the proxy-based degradation detector. For simplicity, we focus on their expected behavior as captured in conditions (i) and (ii) of Theorem 1 and then invoke sub-Gaussian concentration.

**Lemma 3** (Saturation detection)**.** *Under Assumption 1 and condition (i) of Theorem 1, there exists an event $\mathcal{E}_{\text{sat}}$ with $\mathbb{P}(\mathcal{E}_{\text{sat}}) \geq 1 - O(\delta_T)$ such that on $\mathcal{E}_{\text{sat}}$ the following holds for all arms $i$ and segments $j$:*

- *(*No false saturation in ascending phase*) For all local pulls $n \leq \nu_{i,j}$ within segment $j$, the gain window average $\bar{g}_i^{(\omega)}$ remains above $\lambda_{\text{cost}} - \tau_{\text{sat}}$, so saturation is never erroneously declared.*
- *(*Bounded saturation delay*) Once $n \geq \nu_{i,j}$ and the arm is in the saturation phase of segment $j$, saturation is detected within at most $O(\omega_{\text{sat}})$ additional pulls, i.e., after at most a constant multiple of the window length,* RESETARM$(i)$ *is triggered.*

*Proof.* Within the ascending phase ($n \leq \nu_{i,j}$), condition (i) states that $\mathbb{E}[\bar{g}_i^{(\omega)}] \geq \lambda_{\text{cost}} + 2\tau_{\text{sat}}$. By sub-Gaussianity of $g_{i,t}$ and Hoeffding's inequality for window averages, we can bound the probability that $\bar{g}_i^{(\omega)} \leq \lambda_{\text{cost}} - \tau_{\text{sat}}$ by an exponentially small term in $\omega_{\text{sat}}$. A union bound over all relevant windows yields the first claim with probability at least $1 - O(\delta_T)$.

In the saturation phase ($n \geq \nu_{i,j}$), condition (i) states that $\mathbb{E}[\bar{g}_i^{(\omega)}] \leq \lambda_{\text{cost}} - 2\tau_{\text{sat}}$. Thus, after $O(\omega_{\text{sat}})$ pulls in this phase, the concentration of the window average around its mean ensures that $\bar{g}_i^{(\omega)} \leq \lambda_{\text{cost}} - \tau_{\text{sat}}$ with high probability, triggering RESETARM$(i)$. The precise constant in $O(\omega_{\text{sat}})$ depends on the sub-Gaussian parameters and $\tau_{\text{sat}}$, and the failure probability can again be bounded by $O(\delta_T)$ via a union bound. □

An analogous argument applies to the proxy-based degradation detector.

**Lemma 4** (Degradation detection)**.** *Under Assumption 2 and condition (ii) of Theorem 1, there exists an event $\mathcal{E}_{\text{deg}}$ with $\mathbb{P}(\mathcal{E}_{\text{deg}}) \geq 1 - O(\delta_T)$ such that for all arms $i$ and all consecutive segments $j$ and $j + 1$:*

- *(*No false degradation*) While arm $i$ remains in segment $j$, the difference $\bar{z}_i^{\text{cur}} - \bar{z}_i^{\text{ref}}$ stays below $\tau_{\text{deg}}$, so no erroneous reset is triggered by the proxy-based detector.*
- *(*Bounded detection delay*) After the transition from segment $j$ to $j + 1$, degradation is detected and* RESETARM$(i)$ *is triggered within $O(\omega_{\text{deg}})$ pulls from the new environment.*

*Proof.* Condition (ii) states that the proxy mean increases by at least $4\tau_{\text{deg}}$ across consecutive segments. Before the change, both $\bar{z}_i^{\text{ref}}$ and $\bar{z}_i^{\text{cur}}$ concentrate around $\theta_{i,j}$, so their difference remains well below $\tau_{\text{deg}}$. After the change to segment $j + 1$, the recent part of the window concentrates around $\theta_{i,j+1}$ while the older part still reflects $\theta_{i,j}$. The mean difference is at least $4\tau_{\text{deg}}$, so with $O(\omega_{\text{deg}})$ samples from segment $j + 1$, $\bar{z}_i^{\text{cur}} - \bar{z}_i^{\text{ref}}$ exceeds $\tau_{\text{deg}}$ with high probability. Sub-Gaussian

concentration and a union bound over all windows yield the stated bounds with failure probability $O(\delta_T)$. □

Combining Lemmas 2, 3, and 4 and taking $\delta_T = T^{-2}$, we obtain a global high-probability event

$$\mathcal{E} \;=\; \mathcal{E}_\mu \cap \mathcal{E}_{\text{sat}} \cap \mathcal{E}_{\text{deg}}$$

with $\mathbb{P}(\mathcal{E}) \geq 1 - O(1/T)$. The contribution of $\mathcal{E}^c$ to the expected regret is thus $O(1)$ and can be safely absorbed into the additive term in (3).

### C.3 Regret in Ascending Phases

We now bound the regret incurred when arms are still in their ascending phases. On the high-probability event $\mathcal{E}$, windowed estimates track the ascending-phase averages $\bar{\mu}_{i,j}$ up to the confidence bonus $b_i(t)$ and a constant bias, and saturation/degradation resets are not triggered prematurely.

Fix an arm $i$ and a segment $j$ with gap $\Delta_{i,j} > 0$ as in Assumption 3. Consider only those time steps $t$ such that:

- $J_i(t) = j$ (arm $i$ is in segment $j$),
- arm $i$ is still in the ascending phase of segment $j$ (local pull count $n \leq \nu_{i,j}$), and
- $I_t = i$ (WCA-UCB selects arm $i$).

Let $N_{i,j}^{\text{asc}}$ denote the number of such selections. In each of these rounds $t$, there exists some arm $i_t^*$ achieving $\mu_t^*$ (defined in (6)) and having segment index $J_{i_t^*}(t)$. By Assumption 3,

$$\mu_t^* - \bar{\mu}_{i,j} \;\geq\; \Delta_{i,j}.$$

On the event $\mathcal{E}$, we have for any arm $k$ and time $t$ in its ascending phase:

$$\hat{\mu}_k^{(W)}(t) \;\geq\; \bar{\mu}_{k,J_k(t)} - b_k(t) - C_{\text{bias}},$$
$$\hat{\mu}_k^{(W)}(t) \;\leq\; \bar{\mu}_{k,J_k(t)} + b_k(t) + C_{\text{bias}}.$$

Whenever WCA-UCB selects arm $i$ at time $t$, its index must dominate that of $i_t^*$:

$$I_i(t) \;=\; \hat{\mu}_i^{(W)}(t) + b_i(t) \;\geq\; \hat{\mu}_{i_t^*}^{(W)}(t) + b_{i_t^*}(t).$$

Combining these inequalities yields

$$\begin{aligned}
\bar{\mu}_{i,j} + b_i(t) - (b_i(t) + C_{\text{bias}}) \;&\geq\; \hat{\mu}_i^{(W)}(t) \\
&\geq\; \hat{\mu}_{i_t^*}^{(W)}(t) + b_{i_t^*}(t) - b_i(t) \\
&\geq\; \bar{\mu}_{i_t^*,J_{i_t^*}(t)} - (b_{i_t^*}(t) + C_{\text{bias}}) + b_{i_t^*}(t) - b_i(t) \\
&\geq\; \mu_t^* - C' b_i(t) - 2C_{\text{bias}},
\end{aligned}$$

for some constant $C' > 0$ (since $b_{i_t^*}(t)$ and $b_i(t)$ are of the same order). Rearranging and using $\mu_t^* - \bar{\mu}_{i,j} \geq \Delta_{i,j}$, we obtain

$$\Delta_{i,j} \;\leq\; C' b_i(t) + 2C_{\text{bias}}.$$

For sufficiently large $n_i(t)$, the bonus $b_i(t)$ dominates the bias term, so we can simplify to

$$\Delta_{i,j} \;\lesssim\; C'' b_i(t) \;=\; C'' c \sqrt{\frac{\Lambda_T}{n_i(t)}},$$

for some constant $C'' > 0$. Squaring both sides gives

$$n_i(t) \;\lesssim\; \frac{C''^2 c^2 \Lambda_T}{\Delta_{i,j}^2}.$$

This reasoning is standard in UCB analyses: once the local sample count $n_i(t)$ for a sub-optimal arm/segment pair $(i,j)$ exceeds a threshold proportional to $\Lambda_T/\Delta_{i,j}^2$, the inequality $I_i(t) \geq I_{i_t^*}(t)$ can no longer hold on $\mathcal{E}$, so arm $i$ is not selected again in segment $j$ while both arms are in their ascending phases. Formally, we obtain:

**Lemma 5** (Selections of a sub-optimal arm in ascending phases)**.** *On the event $\mathcal{E}$, for any arm $i$ and segment $j$ with gap $\Delta_{i,j} > 0$, the number of times $N_{i,j}^{\text{asc}}$ that WCA-UCB selects arm $i$ in segment $j$ while it is still in the ascending phase is bounded by*

$$N_{i,j}^{\text{asc}} \;\le\; C_3 \frac{\Lambda_T}{\Delta_{i,j}^2} + 1$$

*for some constant $C_3 > 0$ depending on $c$, $B$, and the bias constants.*

*Proof.* The argument above shows that once $n_i(t)$ exceeds a threshold of order $\Lambda_T/\Delta_{i,j}^2$, the selection condition $I_i(t) \ge I_{i_t^*}(t)$ is violated on $\mathcal{E}$. Thus WCA-UCB can select arm $i$ in the ascending phase of segment $j$ at most that many times. The constant $+1$ accounts for the last possible selection before the threshold is reached, and $C_3$ collects all numerical factors. □

Each such selection incurs at most $\Delta_{i,j}$ regret, so summing over all sub-optimal $(i, j)$ yields the main $\Lambda_T/\Delta_{i,j}^2$ term in Theorem 1.

### C.4 Regret in Saturation and Degraded Periods

We now bound the regret incurred when an arm has already entered its saturation phase or when the environment has degraded but the corresponding reset has not yet been triggered.

**Saturation periods** Fix an arm $i$ and a segment $j$. Once the local pull count exceeds $\nu_{i,j}$, the arm is in the saturation phase of this segment. By Assumption 1(b), its expected net reward is non-positive, while the oracle may still obtain positive reward $\mu_t^* \ge 0$ from other arms. On the high-probability event $\mathcal{E}_{\text{sat}}$, Lemma 3 ensures that saturation is detected and RESETARM$(i)$ is triggered within $O(\omega_{\text{sat}})$ additional pulls. Each of these pulls incurs regret at most some constant $\Delta_{\max}$, defined as an upper bound on the absolute difference between the best and worst expected rewards across all arms and segments. Thus the total regret from saturation periods of segment $j$ of arm $i$ is at most $O(\omega_{\text{sat}}\Delta_{\max})$. Summing over all $(i, j)$ yields an $O(M\omega_{\text{sat}}\Delta_{\max})$ contribution.

**Degraded periods** Analogously, when the environment degrades from segment $j$ to $j + 1$ for arm $i$, the proxy signal changes but the reward dynamics may still be temporarily similar until sufficient evidence accumulates. On the high-probability event $\mathcal{E}_{\text{deg}}$, Lemma 4 guarantees that degradation is detected and RESETARM$(i)$ is triggered within $O(\omega_{\text{deg}})$ pulls from the new environment. During this delay, WCA-UCB may still treat the arm as if it were in the old segment and thus incur additional regret compared to the oracle that has full knowledge of the change-points. Again each such pull contributes at most $\Delta_{\max}$ regret, so the total regret over all degradation events is at most $O(M\omega_{\text{deg}}\Delta_{\max})$.

Collecting the saturation and degradation terms, and noting that $\max\{\omega_{\text{sat}}, \omega_{\text{deg}}\}$ dominates both window lengths, we obtain the additive term $C_2 M \max\{\omega_{\text{sat}}, \omega_{\text{deg}}\}$ in Theorem 1.

### C.5 Completion of the Proof of Theorem 1

We are now ready to combine the above components and prove Theorem 1.

*Proof of Theorem 1.* Choose $\delta_T = T^{-2}$ and set $\Lambda_T = \ln(1/\delta_T) = O(\log T)$. By Lemmas 2, 3, and 4, the global event

$$\mathcal{E} = \mathcal{E}_\mu \cap \mathcal{E}_{\text{sat}} \cap \mathcal{E}_{\text{deg}}$$

holds with probability at least $1 - O(1/T)$. The contribution of $\mathcal{E}^c$ to the expected regret is thus $O(1)$.

Conditioned on $\mathcal{E}$, Lemma 5 implies that for each sub-optimal $(i, j)$, the number of times WCA-UCB selects arm $i$ in the ascending phase of segment $j$ is at most $C_3\Lambda_T/\Delta_{i,j}^2 + 1$. Each such selection incurs regret at most $\Delta_{i,j}$, so the total regret from ascending phases is at most

$$R_{\text{asc}}(T) \;\le\; C_1\Lambda_T \sum_{i=1}^{K} \sum_{j=1}^{M_i} \frac{1}{\Delta_{i,j}^2} \;+\; O(1),$$

for some constant $C_1 > 0$.

From the previous subsection, the regret from saturation and degraded periods is bounded by

$$R_{\text{sat/deg}}(T) \;\le\; C_2 M \max\{\omega_{\text{sat}}, \omega_{\text{deg}}\},$$

for some constant $C_2 > 0$ depending on $\Delta_{\max}$ and the sub-Gaussian parameters.

Summing the contributions and adding the $O(1)$ term from $\mathcal{E}^c$, we obtain the bound

$$R(T) \;\le\; C_1 \Lambda_T \sum_{i=1}^{K} \sum_{j=1}^{M_i} \frac{1}{\Delta_{i,j}^2} \;+\; C_2 M \max\{\omega_{\text{sat}}, \omega_{\text{deg}}\} \;+\; O(1),$$

which is precisely (3). Finally, if $M = O(1)$ and $W, \omega_{\text{sat}}, \omega_{\text{deg}} = O(\log T)$, and we recall that $\Lambda_T = O(\log T)$, then

$$R(T) \;=\; \tilde{O}\left(\frac{\log T}{\Delta_{\min}^2}\right),$$

i.e., WCA-UCB achieves a logarithmic regret bound of the same order as classical UCB in a single-segment stationary setting, even with asynchronous segments and convergence-aware resets. This completes the proof. □

# D Reproducibility

Code is available at https://github.com/myfilezone/WCA-UCB. The repository contains the implementation of WCA-UCB, the non-stationary bandit baselines, the trace-driven simulator, synthetic trace generation scripts, and scripts for reproducing the multi-tenant LoRA fine-tuning prototype.

**Dependencies.** We use Python with PyTorch, Hugging Face Transformers, PEFT, bitsandbytes, datasets, accelerate, NumPy, pandas, and Matplotlib. The full experimental environment is provided in the repository as `requirements_full.txt`. It can be installed via:

```
pip install -r requirements_full.txt
```

The exported environment used in our experiments includes CUDA 12.4 PyTorch packages, Hugging Face Transformers, PEFT, bitsandbytes, datasets, accelerate, and common scientific Python libraries.

**Running WCA-UCB simulations.** The trace-driven experiments use synthetic traces following the ascending-saturating reward model described in Assumption 1. The simulator generates slot-level traces with Gaussian observation noise and arm-specific change-points, and then evaluates WCA-UCB against Random, Round-Robin, Greedy, RAW-UCB, SW-UCB, D-UCB, and GLR-klUCB. The default simulation setting uses $K = 5$ active slots and $T = 10000$ rounds.

**Multi-tenant LoRA prototype.** The prototype uses Qwen2.5-1.5B-Instruct with BF16 LoRA adapters ($r = 8, \alpha = 16$). We construct $N = 20$ shadow tenants from UltraChat, a large-scale instruction-following dialogue corpus, using heterogeneous subsets such as code, math, and creative writing. The edge-like node exposes $K = 5$ train-ready adapter slots, and tenants are multiplexed into slot sessions to induce tenant churn and distribution shifts. The mini-batch training loss is used as a noisy low-overhead proxy for windowed shift detection, while PPL-based metrics are used for offline model-quality evaluation.

**Outputs.** The released scripts save raw logs, summary CSV files, and publication-ready figures under the experiment output directory. The reported prototype metrics include geometric-mean PPL, average proxy loss, and training-target switches, where a switch denotes a change of the selected resident slot between two consecutive update rounds.

**Computational cost.** All LoRA fine-tuning prototype experiments are run on a single NVIDIA RTX 4090 GPU with 24 GB memory. Trace-driven simulations are lightweight and can be run on CPUs or a single GPU. The LoRA prototype experiments are more expensive because they involve repeated incremental adapter updates and periodic PPL evaluation; the exact runtime depends on the number of seeds, rounds, and evaluation interval.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the slot-constrained bandit formulation, the WCA-UCB scheduler, the dynamic regret guarantee, and the simulation/prototype evaluations. The scope is also bounded by the learning-layer scheduler with exogenous system-layer admission and by the stated experimental workloads.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses key scope limitations through the slot-based decoupling in Section 3.1, the modeling assumptions in Section 3.2, and the bounded-change conditions in Section 5. The conclusion further identifies future work on joint admission/scheduling and wall-clock/state-transfer measurements on lower-power edge accelerators.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical result is stated under Assumptions 1–3, with a proof outline in Section 5. The complete proof, including supporting lemmas and concentration arguments, is provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6.1 describes the trace generation process, prototype model, LoRA configuration, tenant construction, baselines, and evaluation metrics. Appendix A provides pseudocode for WCA-UCB, and Appendix B reports ablation and sensitivity experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: An code repository is provided at https://github.com/myfilezone/WCA-UCB. The repository contains the implementation of WCA-UCB, non-stationary bandit baselines, trace-driven simulation code, synthetic trace generation scripts, and the multi-tenant LoRA prototype. Appendix D provides reproduction instructions, dependency information, and the hardware setup. The experiments use public assets, including Qwen2.5-1.5B-Instruct and UltraChat, and the full exported environment is provided as `requirements_full.txt`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6.1 specifies the simulation setting, prototype model, LoRA adapter configuration, number of train-ready slots, tenant construction, baselines, and metrics. The appendices provide additional pseudocode, ablation, and sensitivity details needed to understand the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The trace-driven simulations supporting the regret and quality–switching claims are averaged over 1000 random seeds, and the corresponding curves report the mean behavior with shaded variability bands in the simulation figures. The ablation and sensitivity studies further evaluate robustness across algorithm variants and hyperparameter choices. Prototype LoRA results are reported as workload-level aggregate metrics rather than formal hypothesis tests, due to the higher cost of repeated fine-tuning runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Appendix D reports the hardware used for the LoRA prototype experiments: a single NVIDIA RTX 4090 GPU with 24 GB memory. It also distinguishes lightweight CPU-friendly trace-driven simulations from the more expensive LoRA prototype runs, which require repeated incremental adapter updates and periodic PPL evaluation. The released repository provides the full dependency file and reproduction scripts for the reported experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research is an algorithmic and systems study using public models/datasets and synthetic traces, without human-subject experiments or release of high-risk new models. We have reviewed the NeurIPS Code of Ethics and believe the work conforms to it.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The work has potential positive impact by improving the efficiency and stability of local LLM personalization, which may reduce resource use and support privacy-preserving adaptation. Potential risks include more efficient personalization being used in unsafe or biased deployments; such systems should be paired with appropriate model-use policies, safety filters, and monitoring when deployed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new pretrained language model, generative model, scraped dataset, or other high-risk asset. It uses existing public assets for experimental evaluation and does not redistribute fine-tuned checkpoints.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The experiments use public assets, including Qwen2.5-1.5B-Instruct and the UltraChat instruction-following dialogue corpus. Qwen2.5-1.5B-Instruct is released under Apache-2.0, UltraChat is distributed under the MIT license, and the paper uses these assets for research evaluation without redistributing them.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, model checkpoint, or software package as a new asset. The main contribution is an algorithmic scheduler and its analysis/evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects. It uses synthetic traces and public instruction-following text data for model fine-tuning experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval or equivalent review is not applicable. No new data are collected from participants.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology, namely the slot-constrained bandit formulation, WCA-UCB scheduler, and regret analysis, does not use LLMs as research tools, data generators, or methodological components. Qwen2.5-1.5B-Instruct appears as the target model being personalized and scheduled in the prototype evaluation, which is standard experimental usage rather than LLM-assisted methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.