ONLINE SERVING PROBLEM WITH COSTLY SMALL-MODEL ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Edge-cloud collaboration with large language models offers a promising way to balance model quality and deployment cost: large models provide strong capabilities but incur substantial inference costs, while smaller models are more efficient and can potentially be improved through continuous adaptation. However, existing routing approaches typically treat model capabilities as fixed and only decide which model should serve each request. We study an online serving problem where the learner jointly decides whether to fine-tune a small model and how to serve incoming requests, while adaptation incurs an immediate cost but changes future model performance. We characterize the fundamental difficulty of online adaptation through lower bounds and propose an adaptive algorithm combining convergence detection, optimistic adaptation, and confidence-based serving. The resulting regret guarantee is instance-adaptive and nearly matches the lower bounds. Experiments on edge-cloud LLM serving benchmarks demonstrate that the proposed method effectively exploits adaptation when it improves the small model while avoiding unnecessary fine-tuning when adaptation provides limited benefits.

1 INTRODUCTION

Large language models have achieved remarkable capabilities, but deploying them for every request incurs substantial computational, latency, and monetary costs. A common strategy is to combine a powerful but expensive large model with a cheaper small model, where the small model can be further improved through fine-tuning or distillation using feedback from the large model.

This creates a new online decision problem beyond conventional model routing. Existing routing methods typically assume fixed model capabilities and only decide which model should serve each request. In contrast, adaptation allows the learner to improve the future capability of the small model, introducing a tradeoff between immediate serving performance and the cost of adaptation: over-adaptation wastes resources, while under-adaptation misses potential future gains.

Motivated by this problem, we study an online serving problem with costly small-model adaptation. We consider heterogeneous requests belonging to different clusters, where the benefit of adaptation may vary across clusters. At each round, the learner chooses among three actions: serving with the large model, serving with the current small model, or serving with the large model while fine-tuning the small model. Therefore, the learner must jointly optimize immediate serving decisions and the allocation of adaptation effort across clusters.

The key difficulty is that different clusters may exhibit fundamentally different adaptation behaviors. Some clusters may never benefit sufficiently from fine-tuning, while others may achieve substantial future improvements after adaptation. To characterize this trade-off, we compare the learner with an offline oracle that knows the adaptation dynamics and optimally allocates fine-tuning effort, leading to an instance-dependent adaptation complexity measuring the useful investment required.

We establish lower bounds showing the unavoidable cost of identifying useful adaptation and develop an adaptive algorithm combining convergence detection, optimistic estimation of adaptation benefits, and confidence-based serving. The resulting regret guarantee adapts to the instance-dependent adaptation complexity and is nearly optimal.

We evaluate the algorithm in an online edge–cloud serving scenario, with Qwen3.5-Plus as the cloud model and Qwen3-4B as the local model, on Hendrycks MATH and XSum. On MATH, where

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fine-tuning improves the small model on easier clusters, the algorithm attains the lowest loss among all compared policies. On XSum, where no cluster benefits from adaptation, it matches always-cloud serving.

Our contributions are summarized as follows:

- We formulate an online model-serving problem with costly and heterogeneous small-model adaptation, where adaptation decisions influence future model performance.
- We establish lower bounds characterizing the fundamental difficulty of online adaptation and the necessity of dependence on useful adaptation complexity.
- We propose an adaptive algorithm combining convergence detection, optimistic adaptation estimation, and confidence-based model selection and prove an instance-adaptive regret guarantee that is nearly optimal.
- We validate the algorithm in online edge–cloud serving: it attains the lowest loss on MATH, where adaptation pays off on easier clusters, and matches the large model on XSum.

1.1 Related Work

Bandits with evolving rewards. Classical multi-armed bandit algorithms provide the foundation for sequential decision making under uncertainty, with extensive studies on exploration- exploitation tradeoffs and regret analysis Auer et al. (2002); Bubeck & Cesa-Bianchi (2012); Lattimore & Szepesvári (2019). While classical formulations typically assume stationary reward distributions, a growing body of work studies bandits with evolving rewards. Non-stationary bandit methods consider temporal changes in reward distributions and develop adaptive exploration strategies Besbes et al. (2014); Garivier & Moulines (2011); Trovo et al. (2020). Another line of research studies rested and rotting bandits, where the reward of an arm depends on previous selections Tekin & Liu (2012); Levine et al. (2017); Seznec et al. (2019; 2020). More recently, rising bandits consider settings where allocating additional resources improves future rewards Metelli et al. (2022); Li et al. (2020); Mussi et al. (2023). These works provide important foundations for learning with evolving arms. However, they typically model reward evolution as an inherent property of the arms, whereas our setting allows the learner to actively intervene through fine-tuning, incurring an immediate cost while improving future model performance.

Edge–cloud LLM collaboration. Serving systems pair a local small model with remote large models to trade quality against cost (Moslem & Kelleher, 2026; Varangot-Reille et al., 2026). Predictive routers assign the query before generation (Ding et al., 2024; Ong et al., 2025); cascades and deferred routers let a cheaper model answer first and escalate only when the attempt looks unreliable (Chen et al., 2023; Aggarwal et al., 2024; Chuang et al., 2025). In Selective Deferred Routing (Miao & Fang, 2026), the local SLM always answers and exposes its hidden states, and a lightweight decider either returns that answer or forwards the request once to a remote LLM. Token-level collaboration spends cloud compute only on hard tokens (Hao et al., 2024; Zheng et al., 2025). These systems neither update the local model, so its capability never improves, nor remain valid once it is updated, since the reward they rely on then drifts.

Bandits for LLM deployment. Bandit algorithms have also been applied to LLM deployment (Bouneffouf & Feraud, 2026; Chen et al., 2026). MetaLLM and MixLLM route each query among a pool of LLMs with bandit or contextual-bandit rules over quality and cost (Nguyen et al., 2025; Wang et al., 2025), and Poon et al. (2026) extend this to sequential selection under evolving prompts. Xia et al. (2024) study convergence-aware online model selection, modeling LLMs whose rewards increase with repeated fine-tuning as time-increasing bandits.

2 Problem Setup

We consider an online model-serving problem with costly adaptation of a small model. The system has access to a large model L and a small model m. The large model provides reliable but expensive responses, while the small model is cheaper but can be improved through fine-tuning using feedback from the large model. Each fine-tuning update incurs an additional cost and affects future performance.

Requests arrive sequentially over T rounds. Each request belongs to one of K clusters, where different clusters may have different adaptation dynamics. Let $k_t \in [K]$ denote the cluster identity at round t. For each cluster k, let $T_k = \sum_{t=1}^{T} \mathbf{1}\{k_t = k\}$ denote the number of requests from cluster k. The learner chooses one of three actions:

$$a_t \in \{L, m, F\}.$$

Action L serves the request using the large model, action m serves it using the current small model, and action F serves the request using the large model while fine-tuning the small model on cluster k_t. Thus, the learner must jointly make serving decisions and allocate adaptation effort across clusters.

2.1 Feedback and Loss Model

Let $n_k(t)$ denote the number of fine-tuning updates allocated to cluster k before round t. The expected loss of the small model on cluster k after n updates is denoted by $M_k(n)$. We assume a decreasing-then-converging adaptation curve: for each cluster k, there exists an unknown τ_k such that

$$M_k(n) < M_k(n-1), \quad 1 \le n \le \tau_k,$$

and

$$M_k(n) = M_k(\tau_k), \quad n \ge \tau_k.$$

Each fine-tuning update incurs cost $c > 0$.

The large model has cluster-dependent expected loss $\ell_{L,k} \in [0, 1]$. Let ϵ_t and ξ_t be zero-mean sub-Gaussian noise terms. At round t, after choosing action a_t, the learner observes a noisy loss given by

$$\ell_t = \begin{cases} \ell_{L,k_t} + \epsilon_t, & a_t = L, \\ M_{k_t}(n_{k_t}(t)) + \epsilon_t, & a_t = m, \\ \ell_{L,k_t} + c + \epsilon_t, & a_t = F. \end{cases}$$

For action F, the adaptation state is updated as $n_{k_t}(t+1) = n_{k_t}(t) + 1$, while all other cluster states remain unchanged. In addition, the learner observes a noisy validation estimate of the updated small-model loss:

$$Y_{k_t, n_{k_t}(t)+1} = M_{k_t}(n_{k_t}(t) + 1) + \xi_t.$$

This observation provides information about the small model's performance after adaptation and enables the learner to estimate the cluster-specific learning curve.

2.2 Benchmark and Regret

We compare the learner against an offline oracle benchmark that knows all adaptation dynamics in hindsight. The oracle chooses an adaptation allocation $\mathbf{N}^\star = (N_1^\star, \ldots, N_K^\star)$ and then commits to the better model for each cluster. This train-then-commit form is optimal because moving any fine-tuning update earlier cannot increase its current loss and can only improve future small-model performance.

For an allocation $\mathbf{N} = (N_1, \ldots, N_K)$, the oracle loss is

$$J(\mathbf{N}) = \sum_{k=1}^{K} \left[N_k(\ell_{L,k} + c) + (T_k - N_k) \min\{\ell_{L,k}, M_k(N_k)\} \right].$$

The optimal allocation is $\mathbf{N}^\star \in \arg\min_{\mathbf{N}} J(\mathbf{N})$, and the corresponding loss is $J^\star = J(\mathbf{N}^\star)$.

The useful adaptation complexity is defined as $B^\star = \sum_{k=1}^{K} N_k^\star$. It measures the total amount of adaptation that is beneficial for the given instance. The regret of an online algorithm is

$$\mathrm{Reg}(T) = \mathbb{E}\left[\sum_{t=1}^{T} \bar{\ell}_t(a_t) - J^\star \right].$$

Our goal is to design an algorithm whose regret adapts to the useful adaptation complexity.

3 LOWER BOUND

We establish lower bounds characterizing the intrinsic difficulty of online adaptation. There are two sources of regret: the standard uncertainty in choosing between the large and small models, and the additional difficulty of deciding whether costly adaptation is worthwhile.

3.1 WORST-CASE MINIMAX LOWER BOUND

Without additional structural assumptions on the adaptation dynamics, no online algorithm can achieve uniformly sublinear regret. The intuition is that the learner cannot distinguish between instances where adaptation will eventually improve the small model and instances where adaptation provides little benefit without paying the cost of fine-tuning.

If the learner adapts aggressively, it may suffer unnecessary adaptation costs on instances where the small model cannot become competitive. If the learner avoids adaptation, it may fail to exploit instances where fine-tuning would substantially improve future performance. This ambiguity leads to the following lower bound.

Theorem 1. *There exists a universal constant $c_0 > 0$ such that, for any online algorithm $\mathcal{A}$,*

$$\sup_{\mathcal{I}} \mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq c_0 T.$$

Thus, over the unrestricted instance class, the worst-case regret is necessarily linear.

3.2 BUDGETED MINIMAX LOWER BOUND

To obtain a meaningful instance-dependent characterization, we next restrict attention to instances with bounded useful adaptation.

For a budget level B, consider the instance class $\mathcal{C}_{\leq B} = \{\mathcal{I} : B^\star(\mathcal{I}) \leq B\}$. This class contains all instances whose offline optimal strategy requires at most B useful fine-tuning updates in total, where $B^\star(\mathcal{I})$ measures the instance-dependent amount of beneficial adaptation.

The following result shows that dependence on the useful adaptation budget is unavoidable.

Theorem 2. *For any $B \leq T/2$, consider the instance class $\mathcal{C}_{\leq B}$. For any online algorithm $\mathcal{A}$,*

$$\sup_{\mathcal{I} \in \mathcal{C}_{\leq B}} \mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq \Omega(\sqrt{KT} + B).$$

4 ALGORITHM

We propose an adaptive algorithm that jointly optimizes serving decisions and cluster-specific adaptation. The algorithm balances three objectives: identifying the better serving model, deciding whether additional fine-tuning is beneficial, and avoiding unnecessary adaptation after convergence.

The algorithm consists of three components:

- *Convergence detection*, which identifies whether the small model has stopped improving for a given cluster and prevents unnecessary fine-tuning.
- *Optimistic adaptation*, which estimates the potential marginal improvement of additional fine-tuning and decides whether investing in another update is worthwhile.
- *Confidence-based serving*, which selects between the large model and the current small model using lower confidence bounds on their expected losses.

At each round, the algorithm first checks whether the current cluster has reached convergence. If so, no further adaptation is performed for this cluster, and the learner focuses on selecting the better serving model.

Otherwise, the algorithm evaluates the potential benefit of an additional fine-tuning update. When the optimistic estimated improvement exceeds the adaptation cost, the learner performs action F; otherwise, it selects between the large and small models using the confidence-based serving rule.

Algorithm 1 Adaptive Fine-Tuning with Confidence-based Serving

Input: horizon T, window size ω, threshold λ
Initialize $n_k = 0$ and $\mathrm{conv}_k = 0$ for all clusters $k = 1, \ldots, K$
for $t = 1, \ldots, T$ **do**
  Observe the cluster identity k_t of the current request
  if $\mathrm{conv}_{k_t} = 0$ **then**
    Estimate the marginal adaptation benefit $\overline{v}_{k_t,t}$ and set $p_{k_t,t} = \min\{1, \overline{v}_{k_t,t}/c\}$
    Sample $B_t \sim \mathrm{Bernoulli}(p_{k_t,t})$
    if $B_t = 1$ **then**
      Choose action F and update $n_{k_t} \leftarrow n_{k_t} + 1$
      Collect the validation feedback of the updated small model
      if $n_{k_t} \geq 2\omega$ **then**
        Run the convergence detector for cluster k_t
        if convergence is detected **then**
          Set $\mathrm{conv}_{k_t} = 1$
        end if
      end if
    else
      Compute $\mathrm{LCB}_{L,k_t}(t) = \widehat{\ell}_{L,k_t}(t) - r_{L,k_t}(t)$, $\mathrm{LCB}_{m,k_t}(t) = \widehat{M}_{k_t,t}(n_{k_t}) - r^{M}_{k_t,t}(n_{k_t})$
      Choose $a_t \in \arg\min_{a\in\{L,m\}} \mathrm{LCB}_{a,k_t}(t)$.
    end if
  else
    Compute post-convergence confidence bounds: $\mathrm{LCB}_{L,k_t}(t)$ and $\mathrm{LCB}^{\infty}_{m,k_t}(t)$
    Choose $a_t \in \arg\min_{a\in\{L,m\}}\{\mathrm{LCB}_{L,k_t}(t), \mathrm{LCB}^{\infty}_{m,k_t}(t)\}$
  end if
end for

4.1 Convergence Detection

For each cluster k, the algorithm maintains a convergence detector to determine whether further fine-tuning is no longer beneficial. The detector is activated after at least 2ω fine-tuning updates have been performed on cluster k.

Let n_k denote the current adaptation state of cluster k. The detector compares two consecutive windows of validation observations:

$$W_{k,1} = \{n_k - 2\omega + 1, \ldots, n_k - \omega\}, \qquad W_{k,2} = \{n_k - \omega + 1, \ldots, n_k\}.$$

For each window, the algorithm fits a local linear predictor of the cluster-specific learning curve and evaluates the predicted loss after one additional update. Denote the two predictions by

$$\widehat{M}^{(1)}_k(n_k + 1) \quad \text{and} \quad \widehat{M}^{(2)}_k(n_k + 1).$$

The convergence statistic is defined as

$$D_k = \left| \widehat{M}^{(1)}_k(n_k + 1) - \widehat{M}^{(2)}_k(n_k + 1) \right|.$$

When the cluster is still improving, both windows follow similar decreasing trends and produce similar predictions. After convergence, both windows become approximately flat. However, around the convergence point, the two windows correspond to different regimes, resulting in a larger prediction discrepancy. Therefore, the algorithm declares cluster k converged when $D_k > \lambda$.

The detected convergence state is

$$\widehat{\tau}_k = \inf\{n_k : D_k > \lambda\}.$$

After convergence is detected, the algorithm stops allocating further fine-tuning updates to cluster k and treats the corresponding small model as a stationary model. The serving decision then reduces to selecting between the large model and the converged small model.

4.2 Optimistic Adaptation

Before convergence is detected, the algorithm must decide whether an additional fine-tuning update is worthwhile for the current cluster. Since the true marginal benefit of adaptation is unknown, the algorithm uses an optimistic estimate of the potential improvement.

For cluster k, let $\mathcal{D}_{k,t}^{\mathrm{val}}$ denote the validation observations collected from previous fine-tuning updates. The algorithm fits a local linear approximation to the cluster-specific learning curve:

$$M_k(q) \approx \beta_{k,t} - \alpha_{k,t} q,$$

where $\alpha_{k,t}$ represents the marginal reduction in loss from an additional fine-tuning update. Let $\widehat{\alpha}_{k,t}$ be the least-squares estimate and $r_{k,t}^{\alpha}$ its confidence radius. The optimistic marginal improvement is defined as

$$\overline{v}_{k,t} = \left[\widehat{\alpha}_{k,t} + r_{k,t}^{\alpha}\right]_+, \qquad [x]_+ = \max\{x, 0\}.$$

This quantity provides an upper-confidence estimate of the benefit of one additional adaptation step. The algorithm then chooses the fine-tuning action with probability

$$p_{k,t} = \min\left\{1, \frac{\overline{v}_{k,t}}{c}\right\}.$$

Specifically, if $B_t \sim \mathrm{Bernoulli}(p_{k_t,t})$ equals one, the learner selects action F and fine-tunes the small model on cluster k_t. Otherwise, it proceeds to the serving decision rule. Therefore, clusters with larger potential improvement relative to the adaptation cost are more likely to receive additional fine-tuning, while clusters with limited expected benefit are adapted less aggressively.

4.3 Confidence-based Serving

Before convergence is detected, the performance of the small model depends on the number of fine-tuning updates allocated to the current cluster. Therefore, the small model cannot be treated as a stationary arm, and the learner must estimate its current loss at the corresponding adaptation state.

For each cluster k, the algorithm maintains an estimate of the large-model loss and the state-dependent small-model loss using the available serving and validation feedback. Let

$$\widehat{\ell}_{L,k}(t) \quad \text{and} \quad \widehat{M}_{k,t}(n_k(t))$$

denote the corresponding loss estimates, with confidence radii

$$r_{L,k}(t) \quad \text{and} \quad r_{k,t}^{M}(n_k(t)).$$

The algorithm constructs lower confidence bounds:

$$\mathrm{LCB}_{L,k}(t) = \widehat{\ell}_{L,k}(t) - r_{L,k}(t),$$

and

$$\mathrm{LCB}_{m,k}(t) = \widehat{M}_{k,t}(n_k(t)) - r_{k,t}^{M}(n_k(t)).$$

At a round where the algorithm does not fine-tune, it selects the model with the smaller lower confidence bound:

$$a_t \in \arg\min_{a \in \{L,m\}} \mathrm{LCB}_{a,k_t}(t).$$

The key observation is that the algorithm does not estimate the loss of every adaptation state independently. Instead, observations collected at different states are used jointly to estimate the underlying learning curve, which allows the uncertainty to be controlled.

After convergence is detected for cluster k, the small model is treated as a stationary arm with fixed loss corresponding to the converged state. The algorithm then maintains post-convergence confidence bounds $\mathrm{LCB}_{L,k_t}(t)$ and $\mathrm{LCB}_{m,k}^{\infty}(t)$ for the large model and the converged small model, respectively. The serving decision is made by selecting the model with the smaller lower confidence bound:

$$a_t \in \arg\min_{a \in \{L,m\}} \{\mathrm{LCB}_{L,k_t}(t), \mathrm{LCB}_{m,k_t}^{\infty}(t)\}$$

Table 1: Average loss, average completion tokens, and the percentage allocated to each arm.

	MATH					XSum				
Method	Loss	Tok.	Train	Local	Cloud	Loss	Tok.	Train	Local	Cloud
Ours	0.338	824	10.5	67.0	22.5	0.909	148	40.7	18.5	40.7
3-arm LCB	0.369	905	25.1	42.4	32.5	0.910	149	29.6	22.0	48.4
Random-p + LCB	0.374	957	29.8	19.1	51.1	0.908	145	29.7	13.5	56.8
Always cloud	0.346	984	0.0	0.0	100.0	0.908	144	0.0	0.0	100.0
Always local	0.359	790	0.0	100.0	0.0	0.942	253	0.0	100.0	0.0

4.4 Regret Analysis

We now state the main regret guarantee of the proposed algorithm.

Theorem 3 (Instance-adaptive regret bound)**.** *The expected regret of the proposed algorithm satisfies*

$$\mathrm{Reg}(T) \le \widetilde{O}(\sqrt{KT}) + O(B^\star) + O(K\omega).$$

Proof sketch. For each cluster, we decompose the regret into four components.

Serving regret. The confidence-based serving rule compares the large model with the state-dependent small model before convergence and reduces to a standard two-model confidence-based selection problem after convergence detection. By the serving regret analysis, $\mathrm{Reg}_{\mathrm{serve}} = \widetilde{O}\left(\sum_{k=1}^{K} \sqrt{T_k}\right)$.

Useful adaptation delay. The offline oracle performs $N_k^\star$ useful fine-tuning updates on cluster k. The optimistic adaptation analysis shows that delaying these updates incurs $O(cN_k^\star)$ regret. Summing over clusters gives $\mathrm{Reg}_{\mathrm{adapt}} = O(cB^\star)$.

Extra adaptation. The optimistic adaptation rule may perform additional updates due to estimation uncertainty. Using the optimistic marginal mass bound, $\mathrm{Reg}_{\mathrm{extra}} = \widetilde{O}\left((1+c)\sum_{k=1}^{K}\sqrt{\frac{T_k}{c}}\right)$.

Convergence detection. The convergence detector delays the transition to stationary serving by at most ω states for each cluster. The refined analysis bounds the additional serving regret by $O(K\omega)$.

Combining the above terms gives the stated regret bound. □

5 Experiment

We evaluate the proposed algorithm in an online edge–cloud serving scenario and answer the following research questions:

- **RQ1** How does the proposed algorithm compare with bandit baselines and with always-cloud and always-local serving?
- **RQ2** What are the arm-selection dynamics during learning, and on which cluster does fine-tuning make the small model better than the large model?
- **RQ3** Are the online cluster boundaries stable?
- **RQ4** How does average token entropy compare with other ways of forming clusters?

Setup. The cloud model is Qwen3.5-Plus with thinking mode disabled, and the local model is Qwen3-4B, fine-tuned online with LoRA via GRPO on cloud completions. We use two datasets: Hendrycks MATH (Hendrycks et al., 2021) and XSum (Narayan et al., 2018). Hendrycks MATH consists of competition mathematics problems spanning seven subjects, each paired with a step-by-step solution and a final answer; the reward is exact-match correctness of the final answer with a length penalty. XSum consists of BBC news articles, each paired with a single-sentence abstractive summary; the reward is the token-level F1 between the generated and reference summaries. Serving incurs $\ell = (1-r) + c_{\mathrm{tok}}L$ with reward r, completion length L, and $c_{\mathrm{tok}} = 2 \times 10^{-4}$; a train pull additionally pays $c = 0.1$. Clusters are the running quartiles ($K = 4$) of the mean token entropy of the current local model's completion.

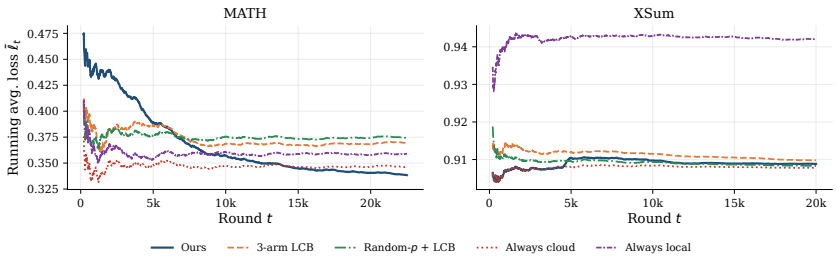


Fig. 1: Running average loss $\bar{\ell}_t$ on MATH (left) and XSum (right).

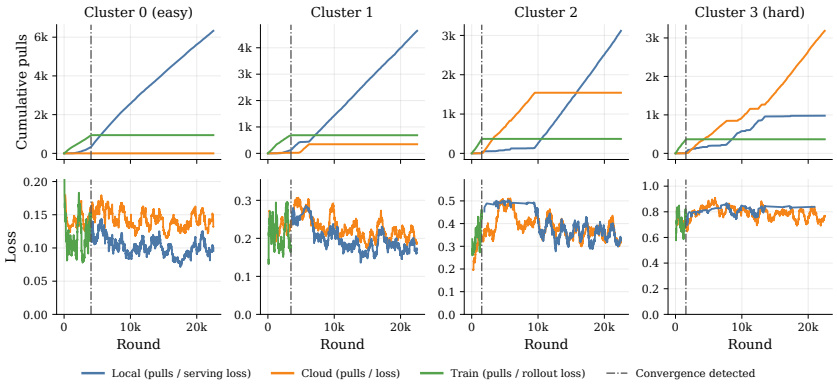


Fig. 2: Per-cluster statistics on MATH. Top: cumulative pulls of each arm; the dash-dotted line marks detected convergence. Bottom: loss of the arm pulls.

Baselines and metric. Baselines include always cloud; always local (without training); 3-arm LCB over {local, cloud, train} without convergence detection or the optimistic improvement test; and Random-p + LCB, which trains with a fixed probability and otherwise serves by LCB. All policies see the same request order, so $t(\bar{\ell}_t^{\pi} - \bar{\ell}_t^{\pi'})$ is the cumulative loss gap between π and π'; the metric ranks policies as regret against a common comparator would, without an estimated oracle.

5.1 RQ1: How does the proposed algorithm compare with the baselines?

MATH. The proposed algorithm attains the lowest average loss of all five policies as shown in Table 1. Its running average starts highest, because early train pulls pay c and local answers come from a model still in training, and then descends monotonically. Both bandit baselines flatten near 0.37, above always-cloud, because without a convergence detector they keep training on 25.1% and 29.8% of rounds (ours: 10.5%) long after training stops paying. Against always-cloud, our algorithm serves only 22.5% of rounds with the cloud, and saves 171 loss units over the stream. It pays a $B^{\star}$-sized discovery cost and then harvests the cheaper local model, as Theorem 3 predicts.

XSum. No cluster admits a local model with lower loss than the cloud, so $B^{\star} = 0$. Always-local is dominated, and the adaptive methods finish within 0.002 of always-cloud. Since a train round serves the cloud completion, the proposed algorithm departs from the cloud line only through local pulls: the small bump in its curve comes from a brief burst of local pulls right after a cluster converges, which confirms that the adapted local model is still worse than the cloud; the algorithm then routes that cluster to the cloud, and its running average returns toward the cloud line. The algorithm matches the large model when adaptation cannot pay and beats it when it can.

5.2 RQ2: What are the arm-selection and fine-tuning dynamics?

Every cluster opens with a steep train segment, and the train curve freezes exactly when convergence is detected (Figure 2, top). The policy then compares two stationary models by LCB. On cluster 0 the

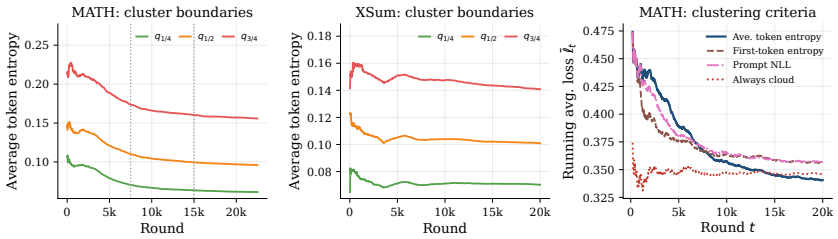


Fig. 3: Left, middle: online quartile boundaries of average token entropy. Right: running average full loss on MATH under three cluster criteria.

small model beats the cloud even before training, since both solve easy problems and the cloud spends more tokens; fine-tuning widens the gap, and the cloud is never pulled. On cluster 1 fine-tuning creates the win, moving the small model from above the cloud to below it; after a short cloud phase the policy commits to local. On cluster 2 the two models are close; the cloud serves the early traffic, and local takes over once its advantage is certified and finishes below the cloud. On cluster 3 fine-tuning helps but not enough; local and cloud share traffic while their confidence intervals overlap, and the cloud then absorbs the cluster. Fine-tuning thus makes the small model better than the large model on clusters 0–2, which are easier by difficulty than cluster 3. On XSum, clusters 0–2 never converge, and cluster 3 converges early and is then served by the cloud. Appendix C gives the XSum arm trace (Figure 4) and per-cluster MATH statistics (Table 3).

5.3 RQ3: Are the online cluster boundaries stable?

On MATH the cluster edges drift from $0.10/0.15/0.22$ to $0.06/0.10/0.16$ during the first epoch as GRPO lowers local entropy, and then flatten; on XSum they settle within $5{,}000$ rounds at $0.07/0.10/0.14$. The three edges move together as the local model becomes more confident, and after that they flatten, remaining strictly ordered and well separated, so each cluster keeps referring to the same slice of traffic.

5.4 RQ4: How does average token entropy compare with other cluster ways?

We replace the clustering score with first-token entropy or prompt NLL and keep everything else fixed. First-token entropy is the entropy of only the first generated token; prompt NLL is the negative log-likelihood of the prompt under the current local model, i.e. how surprising the request looks before any tokens are generated. Average token entropy is the only criterion whose clusters get monotonical with question difficulty and learning potential, because it averages uncertainty over the whole completion and therefore tracks how hard the problem is to solve. First-token entropy does not: the first token is dominated by the local wording of the opening step. Prompt NLL does keep a monotone trend, but a likely prompt is not an easy generation, so easy and hard traffic still share a cluster. In both cases the clusters mix requests that should be trained with requests that should not, the LCB therefore has no arm that is right for an entire cluster and falls back on the cloud, and only average token entropy finishes below always-cloud (Figure 3, right; Figure 5, Appendix C).

6 Conclusion

We studied online model serving with costly and heterogeneous small-model adaptation, where adaptation is an online investment decision: the learner must determine when and where improving the small model is worth the cost. Our algorithm combines optimistic adaptation, confidence-based serving, and convergence detection, and its regret nearly matches the fundamental lower bounds. In edge–cloud experiments, it attains the lowest loss on MATH, where adaptation pays off on the easier clusters, and matches the cloud model on XSum, where it does not; clustering by the local model's average token entropy is stable and aligned with difficulty. Future work includes incorporating richer contextual information and studying how adaptation generalizes across related request clusters.

AI USE STATEMENT

We used generative AI tools as follows, corresponding to the required disclosure categories. We did not use them to generate synthetic data sets. We used them to help develop the theoretical model and conceptual framework of online serving with costly, heterogeneous small-model adaptation, including the three-action decision problem and the instance-dependent useful-adaptation complexity. We used them to formulate mathematical claims and to provide critical ingredients for proving those claims, and they assisted in the writing of proofs in the main text and the appendix. We used them to propose and refine hypotheses about when fine-tuning the small model is worth the cost, to design and provide feedback on the research methodology and experiments (including clustering criteria, baselines, and evaluation metrics), and to implement, modify, and debug parts of the experimental codebase. We did not use them for translation, to clean or reformat a dataset, or to support qualitative or thematic data analysis. We used them to interpret experimental results and to draft and revise wording for clarity and consistency of terminology. We did not use them to generate experimental results, figures, or numerical claims. All AI-assisted text, proofs, and code were reviewed by the authors, who take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Algorithm 1 states the online decision rule. Appendix C lists the models, loss, rewards, clustering rule, and all remaining hyper-parameters. Hendrycks MATH and XSum are public; the MATH stream repeats the $7{,}500$ training problems for three epochs, and the XSum stream presents $20{,}000$ training articles once. Cloud completions are computed with Qwen3.5-Plus, and always-cloud and always-local are evaluated on the same request order as the adaptive methods. Additional per-cluster statistics, the XSum arm trace, and the clustering-criterion ablation are also in Appendix C.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam Mausam. Automix: Automatically mixing language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 131000–131034. Curran Associates, Inc., 2024. doi: 10.52202/079017-4164. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ecda225cb187b40ea8edc1f46b03ffda-Paper-Conference.pdf.

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

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

Djallel Bouneffouf and Raphael Feraud. Multi-armed bandits meet large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 39682–39690, 2026.

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Siguang Chen, Chunli Lv, and Miao Xie. A component-based survey of interactions between large language models and multi-armed bandits. *arXiv preprint arXiv:2601.12945*, 2026.

Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-device llm routing from benchmarking to generalization, 2025. URL https://arxiv.org/abs/2502.04428.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International conference on algorithmic learning theory*, pp. 174–188. Springer, 2011.

Zixu Hao, Huiqiang Jiang, Shiqi Jiang, Ju Ren, and Ting Cao. Hybrid slm and llm for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, EdgeFM '24, pp. 36–41, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706639. doi: 10.1145/3662006.3662067. URL https://doi.org/10.1145/3662006.3662067.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Tor Lattimore and Csaba Szepesvári. Bandit algorithms. *Cambridge University*, 355, 2019.

Nir Levine, Koby Crammer, and Shie Mannor. Rotting bandits. *Advances in neural information processing systems*, 30, 2017.

Yang Li, Jiawei Jiang, Jinyang Gao, Yingxia Shao, Ce Zhang, and Bin Cui. Efficient automatic cash via rising bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4763–4771, 2020.

Alberto Maria Metelli, Francesco Trovo, Matteo Pirola, and Marcello Restelli. Stochastic rising bandits. In *International Conference on Machine Learning*, pp. 15421–15457. PMLR, 2022.

Qijun Miao and Zhixuan Fang. Selective deferred routing: Enabling cost-efficient collaboration between local SLMs and remote LLMs. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yasmin Moslem and John D Kelleher. Dynamic model routing and cascading for efficient llm inference: A survey. *arXiv preprint arXiv:2603.04445*, 2026.

Marco Mussi, Alessandro Montenegro, Francesco Trovò, Marcello Restelli, and Alberto Maria Metelli. Best arm identification for stochastic rising bandits. *arXiv preprint arXiv:2302.07510*, 2023.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018.

Quang H. Nguyen, Thinh Dao, Duy C. Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V. Chawla, and Khoa D. Doan. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms, 2025. URL https://arxiv.org/abs/2407.10834.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Manhin Poon, XiangXiang Dai, Xutong Liu, Fang Kong, John CS Lui, and Jinhang Zuo. Online multi-llm selection via contextual bandits under unstructured context evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24855–24863, 2026.

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*, pp. 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*, pp. 3784–3794. PMLR, 2020.

Cem Tekin and Mingyan Liu. Online learning of rested and restless bandits. *IEEE Transactions on Information Theory*, 58(8):5588–5611, 2012.

Francesco Trovo, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Sliding-window thompson sampling for non-stationary settings. *Journal of Artificial Intelligence Research*, 68:311–364, 2020.

Clovis Varangot-Reille, Christophe Bouvard, Mathieu Ciancone, Antoine Gourru, Marion Schaeffer, and François Jacquenet. Doing more with less: A survey on routing strategies for resource optimisation in large language model-based systems. *J. Artif. Int. Res.*, 86, June 2026. ISSN 1076-9757. doi: 10.1613/jair.1.19801. URL https://doi.org/10.1613/jair.1.19801.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. MixLLM: Dynamic routing in mixed large language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10912–10922, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.545. URL https://aclanthology.org/2025.naacl-long.545/.

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*, pp. 4059–4070, 2024.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P. Xing, Hongyi Wang, and Huaxiu Yao. Citer: Collaborative inference for efficient large language model decoding with token-level routing, 2025. URL https://arxiv.org/abs/2502.01976.

A Lower Bound Proofs

A.1 Proof of Theorem 1

Proof. It suffices to consider the special case with a single cluster, since the class of K-cluster instances contains all single-cluster instances.

We construct two instances $\mathcal{I}_0$ and $\mathcal{I}_1$ that are indistinguishable until a sufficient amount of fine-tuning has been performed, but require different adaptation decisions.

Let the large model have expected loss

$$\ell_L = \frac{1}{2},$$

and let the fine-tuning cost be

$$c = \frac{1}{32}.$$

Define two convergence points

$$\tau_0 = \frac{T}{4}, \qquad \tau_1 = \frac{T}{2}.$$

Consider the common learning curve

$$M(n) = \frac{7}{8} - \frac{n}{T}.$$

For instance $\mathcal{I}_0$, the small model follows this curve until τ_0 and then converges:

$$M_0(n) = \begin{cases} \frac{7}{8} - \frac{n}{T}, & n \le \tau_0, \\ \frac{5}{8}, & n > \tau_0. \end{cases}$$

For instance $\mathcal{I}_1$, the small model follows the same curve until τ_1 and then converges:

$$M_1(n) = \begin{cases} \frac{7}{8} - \frac{n}{T}, & n \le \tau_1, \\ \frac{3}{8}, & n > \tau_1. \end{cases}$$

Hence, the two instances have identical observations before τ_0 fine-tuning updates:

$$M_0(n) = M_1(n), \qquad n \le \tau_0.$$

Therefore, before the learner has performed τ_0 fine-tuning updates, it cannot distinguish whether the small model has already converged or whether further fine-tuning will continue to improve its performance.

We first characterize the offline optimal behavior on the two instances. On $\mathcal{I}_0$, the converged small model satisfies

$$M_0(\tau_0) = \frac{5}{8} > \frac{1}{2} = \ell_L.$$

Thus, further fine-tuning cannot make the small model competitive with the large model, and the optimal strategy performs no adaptation.

On $\mathcal{I}_1$, after τ_1 fine-tuning updates, the small model reaches

$$M_1(\tau_1) = \frac{3}{8} < \frac{1}{2} = \ell_L.$$

The total future improvement obtained from adaptation is

$$(T - \tau_1)(\ell_L - M_1(\tau_1)) = \frac{T}{2} \cdot \frac{1}{8} = \frac{T}{16},$$

whereas the total fine-tuning cost is

$$c\tau_1 = \frac{1}{32} \cdot \frac{T}{2} = \frac{T}{64}.$$

Therefore, the benefit of adaptation exceeds its cost, and useful adaptation is necessary in this instance.

We consider two cases.

First, suppose that under instance $\mathcal{I}_0$ the algorithm reaches τ_0 fine-tuning updates with non-negligible probability. Since the two instances are indistinguishable before this point, the algorithm cannot know whether it is in $\mathcal{I}_0$ or $\mathcal{I}_1$. On $\mathcal{I}_0$, however, all these updates are unnecessary because the small model never becomes better than the large model. Therefore, each additional update only incurs adaptation cost, and the expected regret satisfies

$$\mathbb{E}_{\mathcal{I}_0}[\mathrm{Reg}_{\mathcal{A}}(T)] = \Omega(c\tau_0) = \Omega(T).$$

Otherwise, the algorithm avoids performing τ_0 fine-tuning updates with high probability. Because the observations before τ_0 updates are identical under $\mathcal{I}_0$ and $\mathcal{I}_1$, the same behavior occurs on $\mathcal{I}_1$. Consequently, the algorithm fails to invest sufficiently early to distinguish whether further adaptation is beneficial. On $\mathcal{I}_1$, the offline optimum eventually obtains a constant loss improvement

$$\ell_L - M_1(\tau_1) = \frac{1}{8},$$

while the learner avoids reaching the useful adaptation regime. This yields

$$\mathbb{E}_{\mathcal{I}_1}[\mathrm{Reg}_{\mathcal{A}}(T)] = \Omega(T).$$

Combining the two cases, for every online algorithm $\mathcal{A}$, there exists an instance $\mathcal{I} \in \{\mathcal{I}_0, \mathcal{I}_1\}$ such that

$$\mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] = \Omega(T).$$

Therefore, there exists a universal constant $c_0 > 0$ such that

$$\sup_{\mathcal{I}} \mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \ge c_0 T.$$

□

A.2 Proof of Theorem 2

Proof. We prove the result by establishing two separate lower bounds over the class $\mathcal{C}_{\leq B}$.

First, consider the difficulty of model selection without useful adaptation. We construct a subclass of instances in $\mathcal{C}_{\leq B}$ with

$$B^\star(\mathcal{I}) = 0.$$

In these instances, the small model is never improved by fine-tuning, and the learner only needs to decide between the large and small models for each cluster.

For each cluster k, the problem reduces to a two-arm stochastic bandit with T_k observations. By the standard minimax lower bound for two-arm bandits, any algorithm suffers regret

$$\Omega(\sqrt{T_k})$$

on cluster k. Summing over all clusters gives

$$\mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq \Omega\left(\sum_{k=1}^{K} \sqrt{T_k}\right) \geq \Omega(\sqrt{KT}).$$

We next prove the unavoidable adaptation cost. Consider the subclass of instances with a single active cluster. We construct two instances using the same indistinguishability argument as in Theorem 1.

In the first instance, the small model converges early and remains inferior to the large model. Hence, the optimal adaptation budget is zero:

$$B^\star(\mathcal{I}_0) = 0.$$

In the second instance, the small model follows the same learning curve until the first convergence point but continues improving afterwards. After B additional fine-tuning updates, it becomes better than the large model. Thus,

$$B^\star(\mathcal{I}_1) = B.$$

Before performing these B additional fine-tuning updates, the two instances are statistically indistinguishable. Therefore, the learner faces the same dilemma as in Theorem 1: if it invests heavily in fine-tuning, it suffers unnecessary adaptation cost on $\mathcal{I}_0$; if it avoids adaptation, it fails to exploit the improved small model on $\mathcal{I}_1$.

Consequently, for any algorithm $\mathcal{A}$, there exists one of the two instances such that

$$\mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq \Omega(B).$$

Since both instances satisfy

$$B^\star(\mathcal{I}) \leq B,$$

they belong to $\mathcal{C}_{\leq B}$, and hence

$$\sup_{\mathcal{I} \in \mathcal{C}_{\leq B}} \mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq \Omega(B).$$

Combining the two lower bounds, we have

$$\sup_{\mathcal{I} \in \mathcal{C}_{\leq B}} \mathbb{E}_{\mathcal{I}}[\mathrm{Reg}_{\mathcal{A}}(T)] \geq \max\{\Omega(\sqrt{KT}), \Omega(B)\}.$$

□

B Additional Algorithm Details and Proofs

In the theoretical analysis, we assume a linear-then-flat structure for the learning curve, which is a commonly used model for diminishing adaptation effects Xia et al. (2024). Specifically, for each cluster k, the small-model loss follows

$$M_k(n) = \beta_k - \alpha_k n, \qquad n < \tau_k,$$

where $\alpha_k > 0$ denotes the marginal improvement rate of fine-tuning before convergence. After convergence, the loss remains constant:

$$M_k(n) = M_k(\tau_k), \qquad n \geq \tau_k.$$

B.1 Convergence Detection Analysis

For each window $W_{k,j}$, $j \in \{1, 2\}$, the detector fits a linear model using ordinary least squares (OLS) over the observations in the corresponding window:

$$(\widehat{\beta}_{k,j}, \widehat{\alpha}_{k,j}) = \arg\min_{\beta,\alpha} \sum_{n \in W_{k,j}} \left(Y_{k,n} - (\beta - \alpha n)\right)^2 .$$

The resulting prediction is

$$\widehat{M}_k^{(j)}(n) = \widehat{\beta}_{k,j} - \widehat{\alpha}_{k,j} n.$$

The one-step prediction is

$$\widehat{M}_k^{(j)}(n_k + 1) = \widehat{\beta}_{k,j} - \widehat{\alpha}_{k,j}(n_k + 1).$$

Lemma 1 (Accuracy of convergence detection)**.** *For each cluster k, let $\widehat{\tau}_k$ denote the convergence index detected by the fixed-window procedure. There exists a choice of threshold λ such that, with probability at least $1 - \delta$,*

$$\tau_k \le \widehat{\tau}_k \le \tau_k + \omega.$$

Proof. The convergence detector is applied independently to each cluster. Since the adaptation dynamics and validation observations are cluster-specific, it suffices to analyze an arbitrary fixed cluster k. For notational simplicity, we omit the cluster index k throughout the proof.

Let the validation observation after the n-th fine-tuning update be

$$Y_n = M(n) + \xi_n,$$

where $\{\xi_n\}$ are zero-mean 1-sub-Gaussian noises.

For a window

$$W = \{a, a + 1, \ldots, a + \omega - 1\},$$

let

$$\bar{j}_W = \frac{1}{\omega} \sum_{j \in W} j$$

and

$$S_W = \sum_{j \in W} (j - \bar{j}_W)^2.$$

For consecutive integer design points,

$$S_W = \frac{\omega(\omega^2 - 1)}{12}.$$

The least-squares prediction at a point x can be written as

$$\widehat{M}_W(x) = \sum_{j \in W} w_j(x) Y_j,$$

where

$$w_j(x) = \frac{1}{\omega} + \frac{(x - \bar{j}_W)(j - \bar{j}_W)}{S_W}.$$

The corresponding noiseless prediction is

$$M_W(x) = \sum_{j \in W} w_j(x) M(j),$$

and therefore

$$\widehat{M}_W(x) - M_W(x) = \sum_{j \in W} w_j(x) \xi_j.$$

Moreover,

$$\sum_{j\in W} w_j(x)^2 = \frac{1}{\omega} + \frac{(x-\bar{j}_W)^2}{S_W}.$$

In the detector, the prediction point is always $x = n_t + 1$. For both windows, the distance between the prediction point and the window center is $O(\omega)$. Therefore, there exists a universal constant C_0 such that

$$\sum_{j\in W_i(t)} w_j(n_t+1)^2 \le \frac{C_0}{\omega}, \qquad i \in \{1,2\}.$$

Since the validation noise is sub-Gaussian, the prediction error is also sub-Gaussian. By a union bound over all detector times and both windows, with probability at least $1-\delta$, the event

$$\mathcal{E} = \left\{ \left| \widehat{M}_t^{(i)}(n_t+1) - M_t^{(i)}(n_t+1) \right| \le \varepsilon_\omega,\ \forall t,\ i \in \{1,2\} \right\}$$

holds, where

$$\varepsilon_\omega = C\sqrt{\frac{\log(T/\delta)}{\omega}}.$$

We condition on the event $\mathcal{E}$.

No false detection before convergence. Consider any detector time satisfying

$$n_t \le \tau.$$

Both windows are contained in the linear regime. Therefore,

$$M_t^{(1)}(n_t+1) = M_t^{(2)}(n_t+1) = \beta - \alpha(n_t+1).$$

Hence,

$$\begin{aligned} D_t &= \left| \widehat{M}_t^{(1)}(n_t+1) - \widehat{M}_t^{(2)}(n_t+1) \right| \\ &\le \left| \widehat{M}_t^{(1)}(n_t+1) - M_t^{(1)}(n_t+1) \right| + \left| \widehat{M}_t^{(2)}(n_t+1) - M_t^{(2)}(n_t+1) \right| \\ &\le 2\varepsilon_\omega. \end{aligned}$$

Therefore, choosing

$$\lambda > 2\varepsilon_\omega$$

prevents false detection before convergence, and thus

$$\widehat{\tau} \ge \tau.$$

Detection within one window after convergence. Consider the first detector time satisfying

$$n_t = \tau + \omega.$$

Then

$$W_1(t) = \{\tau - \omega + 1, \dots, \tau\}$$

lies in the linear regime, while

$$W_2(t) = \{\tau + 1, \dots, \tau + \omega\}$$

lies in the flat regime.

The prediction point is

$$n_t + 1 = \tau + \omega + 1.$$

For the first window,

$$M_t^{(1)}(\tau+\omega+1) = \beta - \alpha(\tau+\omega+1),$$

while for the second window,

$$M_t^{(2)}(\tau+\omega+1) = M(\tau) = \beta - \alpha\tau.$$

Therefore, the separation between the two noiseless predictions is

$$\left|M_t^{(1)}(\tau+\omega+1) - M_t^{(2)}(\tau+\omega+1)\right| = \alpha(\omega+1).$$

Using the event $\mathcal{E}$,

$$D_t \geq \alpha(\omega+1) - 2\varepsilon_\omega.$$

Thus, if

$$\lambda < \alpha(\omega+1) - 2\varepsilon_\omega,$$

the detector must trigger no later than this time, yielding

$$\widehat{\tau} \leq \tau + \omega.$$

Combining the two parts, choosing

$$2\varepsilon_\omega < \lambda < \alpha(\omega+1) - 2\varepsilon_\omega$$

gives

$$\tau \leq \widehat{\tau} \leq \tau + \omega.$$

□

B.2 Optimistic Adaptation Analysis

This section provides the detailed implementation and theoretical analysis of the optimistic adaptation procedure. Since the adaptation process is cluster-specific, we analyze an arbitrary cluster and omit the cluster index for simplicity.

B.2.1 Learning Curve Estimation

After each fine-tuning update at state n, the learner receives a validation observation

$$Y_n = M(n) + \xi_n,$$

where ξ_n is a zero-mean 1-sub-Gaussian noise.

Therefore, before convergence,

$$Y_n = \beta - \alpha n + \xi_n.$$

Let

$$\mathcal{D}_t^{\text{val}} = \{(q_i, Y_i)\}_{i=0}^{n_t}$$

denote the validation observations collected before round t. Since a validation observation is obtained after every fine-tuning update, each visited adaptation state has an associated observation.

The algorithm estimates the learning curve parameters by least squares:

$$(\widehat{\beta}_t, \widehat{\alpha}_t) = \arg\min_{\beta,\alpha} \sum_{i=0}^{n_t} (Y_i - \beta + \alpha q_i)^2.$$

The estimated marginal improvement rate is

$$\widehat{\alpha}_t = -\frac{\sum_{i=0}^{n_t}(q_i - \bar{q}_t)(Y_i - \bar{Y}_t)}{\sum_{i=0}^{n_t}(q_i - \bar{q}_t)^2},$$

where

$$\bar{q}_t = \frac{1}{n_t+1}\sum_{i=0}^{n_t} q_i,$$

and

$$\bar{Y}_t = \frac{1}{n_t+1}\sum_{i=0}^{n_t} Y_i.$$

The confidence radius of the estimated marginal improvement is defined as

$$r_t^\alpha = \sqrt{\frac{2\log(2T/\delta)}{\sum_{i=0}^{n_t}(q_i - \bar{q}_t)^2}}.$$

For consecutive fine-tuning states

$$q_i = i, \qquad i = 0, \dots, n_t,$$

we have

$$\sum_{i=0}^{n_t}(q_i - \bar{q}_t)^2 = \frac{n_t(n_t+1)(n_t+2)}{12}.$$

Hence,

$$r_t^\alpha \le C\sqrt{\log(T/\delta)}\frac{1}{n_t^{3/2}}.$$

Define the confidence event

$$\mathcal{E}_{\mathrm{ft}} = \{|\widehat{\alpha}_t - \alpha| \le r_t^\alpha,\ \forall t\}.$$

By the concentration inequality for least-squares estimation,

$$\mathbb{P}(\mathcal{E}_{\mathrm{ft}}) \ge 1 - \delta.$$

On the event $\mathcal{E}_{\mathrm{ft}}$, the optimistic estimate

$$\overline{v}_t = [\widehat{\alpha}_t + r_t^\alpha]_+$$

satisfies

$$\alpha \le \overline{v}_t.$$

The probability of performing an additional fine-tuning update is

$$p_t = \min\left\{1, \frac{\overline{v}_t}{c}\right\}.$$

B.2.2 Useful Fine-Tuning Delay

Lemma 2 (Useful fine-tuning delay)**.** *On the confidence event $\mathcal{E}_{\mathrm{ft}}$, the expected regret caused by delaying the $N^\star$ useful fine-tuning updates selected by the offline benchmark satisfies*

$$\mathrm{Reg}_{\mathrm{delay}} \le O(cN^\star).$$

Proof. Consider the j-th useful fine-tuning update performed by the offline benchmark. Before the learner performs this update, the learner may be one fine-tuning step behind. The resulting loss increase is at most the marginal improvement:

$$v(j) = M(j) - M(j+1).$$

On the confidence event,

$$v(j) \le \overline{v}_j.$$

The probability of performing this update at a given opportunity is

$$p_j = \min\left\{1, \frac{\overline{v}_j}{c}\right\}.$$

If $\overline{v}_j > c$, then the update is performed immediately.

Otherwise,

$$p_j = \frac{\overline{v}_j}{c}.$$

The expected regret caused by waiting for this update is bounded by

$$\frac{\overline{v}_j}{p_j} \le c.$$

Therefore, each useful fine-tuning update contributes at most $O(c)$ regret. Summing over the $N^\star$ useful updates gives

$$\mathrm{Reg}_{\mathrm{delay}} \le O(cN^\star).$$

□

B.2.3 Extra Fine-Tuning

Let A denote the number of extra fine-tuning updates performed after the benchmark stopping point. Define the remaining optimistic marginal mass as

$$V^+ = \sum_{n \ge N^\star} \overline{v}(n).$$

Lemma 3 (Extra fine-tuning bound)**.** *On the confidence event $\mathcal{E}_{\mathrm{ft}}$,*

$$\mathbb{E}[A] \le \sqrt{\frac{TV^+}{c}}.$$

Consequently, the regret caused by extra fine-tuning is bounded by

$$\widetilde{O}\left((1+c)\sqrt{\frac{TV^+}{c}}\right)$$

Proof. Index the extra fine-tuning updates by $j = 1, 2, \ldots$. The probability of performing the j-th update is

$$p_j = \min\left\{1, \frac{\overline{v}_j}{c}\right\}.$$

Define the potential function

$$\Phi(m) = \sum_{j=1}^{m} \frac{1}{p_j}.$$

Before an update is triggered, the expected increase of Φ in one round is at most one because the update occurs with probability p_j. Therefore,

$$\mathbb{E}[\Phi(A)] \le T.$$

Moreover,

$$\frac{1}{p_j} \ge \frac{c}{\overline{v}_j}.$$

By Cauchy–Schwarz,

$$\sum_{j=1}^{m} \frac{1}{\overline{v}_j} \ge \frac{m^2}{\sum_{j=1}^{m} \overline{v}_j}.$$

Thus,

$$\Phi(m) \ge \frac{cm^2}{V^+}.$$

Taking $m = A$ gives

$$\mathbb{E}[A^2] \le \frac{TV^+}{c}.$$

Since $A \ge 0$,

$$\mathbb{E}[A] \le \sqrt{\mathbb{E}[A^2]} \le \sqrt{\frac{TV^+}{c}}.$$

Each extra fine-tuning update incurs at most $O(1+c)$ regret, completing the proof. □

B.2.4 Bounding the Optimistic Marginal Mass

Lemma 4 (Optimistic marginal mass)**.** *The optimistic marginal mass V^+ satisfies:*

$$V^+ \le 1 + C\sqrt{\log(T/\delta)} \sum_{n=2}^{\tau} \frac{1}{n^{3/2}}.$$

Consequently,

$$V^+ = \widetilde{O}(1).$$

Proof. The true marginal improvement is

$$v(n) = M(n) - M(n+1).$$

Since the loss is bounded in $[0, 1]$,

$$\sum_{n=0}^{\tau-1} v(n) = M(0) - M(\tau) \le 1.$$

On the confidence event,

$$\overline{v}(n) \le v(n) + 2r_n^{\alpha}.$$

Therefore,

$$\begin{aligned} V^+ &\le \sum_{n=0}^{\tau-1} v(n) + 2\sum_{n=2}^{\tau} r_n^{\alpha} + O(1) \\ &\le 1 + C\sqrt{\log(T/\delta)} \sum_{n=2}^{\tau} \frac{1}{n^{3/2}}. \end{aligned}$$

Since

$$\sum_{n=2}^{\infty} n^{-3/2} < \infty,$$

we obtain

$$V^+ = \widetilde{O}(1).$$

□

B.3 Confidence-based Serving Analysis

We provide the detailed implementation and theoretical analysis of the confidence-based serving procedure. The analysis is performed uniformly over all clusters. For each cluster k, the learner maintains separate statistics and confidence bounds.

The serving procedure has two phases. Before convergence is detected, the small-model loss depends on the current adaptation state and is estimated using the underlying learning curve. After convergence is detected, the small model becomes stationary and the serving problem reduces to a standard two-model selection problem.

B.3.1 Before Covergence Detection

Before convergence, we write the small model learning curve:

$$M_k(n) = x(n)^{\top}\theta_k^{\star},$$

where

$$x(n) = \begin{pmatrix} 1 \\ -n \end{pmatrix}, \qquad \theta_k^{\star} = \begin{pmatrix} \beta_k \\ \alpha_k \end{pmatrix}.$$

Using all available small-model observations from cluster k, the learner constructs

$$V_{k,t} = \rho I + \sum_i x(q_i)x(q_i)^\top,$$

and estimates

$$\widehat{\theta}_{k,t} = V_{k,t}^{-1} \sum_i x(q_i) Z_i,$$

where Z_i is the corresponding noisy observation of the small-model loss.

The predicted small-model loss is

$$\widehat{M}_{k,t}(n) = x(n)^\top \widehat{\theta}_{k,t}.$$

The confidence radius is

$$r_{k,t}^M(n) = \gamma_{k,t} \sqrt{x(n)^\top V_{k,t}^{-1} x(n)},$$

where

$$\gamma_{k,t} = \sigma \sqrt{2 \log \frac{\det(V_{k,t})^{1/2}}{\det(\rho I)^{1/2} \delta/(3K)}} + \sqrt{\rho} S.$$

Here, $\rho > 0$ is the regularization parameter, and S is a sufficiently large constant.

For each cluster k, the large model has a stationary loss $\ell_{L,k}$. Whenever the learner serves a request using the large model or performs fine-tuning on cluster k, it observes

$$Z_{L,k,t} = \ell_{L,k} + \epsilon_{L,k,t},$$

where $\epsilon_{L,k,t}$ is a zero-mean sub-Gaussian noise.

Let

$$N_{L,k}(t) = \sum_{s<t} \mathbf{1}\{k_s = k,\ a_s \in \{L, F\}\}$$

denote the number of large-model observations collected from cluster k before round t.

The empirical estimate of the large-model loss is

$$\widehat{\ell}_{L,k}(t) = \frac{1}{N_{L,k}(t) \vee 1} \sum_{\substack{s<t: \\ k_s=k,\ a_s \in \{L,F\}}} Z_{L,k,s}.$$

The corresponding confidence radius is

$$r_{L,k}(t) = \sqrt{\frac{2 \log(6KT/\delta)}{N_{L,k}(t) \vee 1}}.$$

The lower confidence bounds are

$$\mathrm{LCB}_{L,k}(t) = \widehat{\ell}_{L,k}(t) - r_{L,k}(t),$$

and

$$\mathrm{LCB}_{m,k}(t) = \widehat{M}_{k,t}(n_k(t)) - r_{k,t}^M(n_k(t)).$$

The learner chooses

$$a_t \in \arg \min_{a \in \{L,m\}} \mathrm{LCB}_{a,k_t}(t).$$

B.3.2 AFTER CONVERGENCE DETECTION

After convergence is detected for cluster k, the small model is treated as a stationary model.

Whenever the learner serves a request using the small model, let $Z_{k,t}^m$ denote the corresponding loss observation.

Let

$$N_{m,k}^{\infty}(t) = \sum_{s<t} \mathbf{1}\{k_s = k,\ a_s = m,\ \mathrm{conv}_k = 1\}$$

denote the number of post-convergence small-model observations collected from cluster k before round t.

The empirical estimate of the converged small-model loss is

$$\widehat{\ell}_{m,k}^{\infty}(t) = \frac{1}{N_{m,k}^{\infty}(t) \vee 1} \sum_{\substack{s<t: \\ k_s=k,\ a_s=m,\ \mathrm{conv}_k=1}} Z_{k,s}^m.$$

The confidence radius is

$$r_{m,k}^{\infty}(t) = \sqrt{\frac{2\log(6KT/\delta)}{N_{m,k}^{\infty}(t) \vee 1}}.$$

The post-detection lower confidence bounds are

$$\mathrm{LCB}_{L,k}(t) = \widehat{\ell}_{L,k}(t) - r_{L,k}(t),$$

and

$$\mathrm{LCB}_{m,k}^{\infty}(t) = \widehat{\ell}_{m,k}^{\infty}(t) - r_{m,k}^{\infty}(t).$$

The learner selects the model with smaller LCB.

B.3.3 CONFIDENCE EVENTS

For each cluster k, define three confidence events.

The large-model confidence event is

$$\mathcal{E}_{L,k} = \left\{ |\widehat{\ell}_{L,k}(t) - \ell_{L,k}| \le r_{L,k}(t),\ \forall t \right\}.$$

Before convergence, the small-model prediction confidence event is

$$\mathcal{E}_{m,k} = \left\{ |\widehat{M}_{k,t}(n) - M_k(n)| \le r_{k,t}^{M}(n),\ \forall t, n \right\}.$$

After convergence, the stationary small-model confidence event is

$$\mathcal{E}_{\infty,k} = \left\{ |\widehat{\ell}_{m,k}^{\infty}(t) - M_k(\tau_k)| \le r_{m,k}^{\infty}(t),\ \forall t \right\}.$$

By concentration inequalities and a union bound, the global confidence event

$$\mathcal{E}_{\mathrm{serve}} = \bigcap_{k=1}^{K} (\mathcal{E}_{L,k} \cap \mathcal{E}_{m,k} \cap \mathcal{E}_{\infty,k})$$

satisfies

$$\mathbb{P}(\mathcal{E}_{\mathrm{serve}}) \ge 1 - \delta.$$

B.3.4 SERVING REGRET

The following lemma establishes the regret bound for the confidence-based serving procedure.

Lemma 5 (Confidence-based serving regret)**.** *Condition on the event in Lemma 1. For each cluster k, the expected regret incurred by the confidence-based serving procedure satisfies*

$$\mathrm{Reg}_{\mathrm{serve},k} \le \widetilde{O}(\sqrt{T_k} + \omega).$$

Consequently,

$$\mathrm{Reg}_{\mathrm{serve}} \le \widetilde{O}\left(\sqrt{KT} + K\omega\right).$$

Proof. We first analyze the regret on the global confidence event $\mathcal{E}_{\mathrm{serve}}$.

The complement event has probability at most δ. Since the per-round loss is bounded, its contribution to the expected regret is at most $O(\delta T)$. Taking $\delta = 1/T$ makes this term negligible and absorbed into the logarithmic factors.

We therefore focus on the event $\mathcal{E}_{\mathrm{serve}}$.

We analyze one cluster k and omit the cluster index for simplicity. The serving regret is decomposed into three phases: (i) before the true convergence point, (ii) between the true convergence point and the detected convergence point, and (iii) after convergence detection.

Before true convergence. For $t < \tau$, the small-model loss is state-dependent and follows the linear learning curve. The algorithm selects the model with the smaller lower confidence bound.

On $\mathcal{E}_{\mathrm{serve}}$, a suboptimal serving decision can only occur due to confidence uncertainty. Therefore, the instantaneous regret satisfies

$$\Delta_t \le 2r_L(t) + 2r_t^M(n_t).$$

For the large model, standard concentration gives

$$\sum_{t<\tau} r_L(t) = \widetilde{O}(\sqrt{T_k}).$$

For the small model,

$$r_t^M(n_t) = \gamma_t \sqrt{x(n_t)^\top V_t^{-1} x(n_t)}.$$

The elliptical potential lemma gives

$$\sum_{t<\tau} \gamma_t \sqrt{x(n_t)^\top V_t^{-1} x(n_t)} = \widetilde{O}(\sqrt{T_k}).$$

Hence, the pre-convergence serving regret is

$$\widetilde{O}(\sqrt{T_k}).$$

After true convergence but before detection. By the convergence detection lemma,

$$\tau \le \widehat{\tau} \le \tau + \omega.$$

Since losses are bounded in $[0, 1]$, the regret in each delayed round is bounded by a constant, giving the alternative bound

$$O(\omega).$$

After convergence detection. After $t \geq \widehat{\tau}$, the small model is stationary:

$$M(n) = M(\tau).$$

The serving problem reduces to selecting between two stationary models, namely the large model and the converged small model. Applying the standard LCB analysis gives $\widetilde{O}(\sqrt{T_k})$ regret.

Combining the three phases yields

$$\mathrm{Reg}_{\mathrm{serve},k} \leq \widetilde{O}(\sqrt{T_k} + \omega).$$

Finally, summing over clusters gives

$$\mathrm{Reg}_{\mathrm{serve}} \leq \widetilde{O}\left(\sqrt{KT} + K\omega\right).$$

□

B.4 Proof of Theorem 3

Proof. We decompose the global regret into cluster-wise regrets:

$$\mathrm{Reg}(T) = \sum_{k=1}^{K} \mathrm{Reg}_k(T_k).$$

For each cluster k, we further decompose

$$\mathrm{Reg}_k = \mathrm{Reg}_{\mathrm{serve},k} + \mathrm{Reg}_{\mathrm{adapt},k} + \mathrm{Reg}_{\mathrm{extra},k} + \mathrm{Reg}_{\mathrm{det},k}.$$

Serving regret. By Lemma 5, the confidence-based serving procedure satisfies

$$\mathrm{Reg}_{\mathrm{serve},k} \leq \widetilde{O}(\sqrt{T_k} + \omega).$$

Therefore,

$$\mathrm{Reg}_{\mathrm{serve}} \leq \widetilde{O}\left(\sqrt{KT} + K\omega\right).$$

Useful adaptation. By Lemma 2, delaying the useful fine-tuning updates chosen by the offline oracle contributes at most

$$\mathrm{Reg}_{\mathrm{adapt},k} \leq O(cN_k^\star).$$

Summing over clusters gives

$$\mathrm{Reg}_{\mathrm{adapt}} \leq O\left(c\sum_{k=1}^{K} N_k^\star\right) = O(cB^\star).$$

Extra adaptation. By Lemma 3, the regret caused by extra fine-tuning satisfies

$$\mathrm{Reg}_{\mathrm{extra},k} \leq \widetilde{O}\left((1+c)\sqrt{\frac{T_k V_k^+}{c}}\right)$$

By Lemma 4,

$$V_k^+ = \widetilde{O}(1).$$

Hence,

$$\mathrm{Reg}_{\mathrm{extra}} \leq \widetilde{O}\left((1+c)\sum_{k=1}^{K}\sqrt{\frac{T_k}{c}}\right).$$

Convergence detection delay. By the convergence detection lemma, for every cluster k,

$$\tau_k \le \widehat{\tau}_k \le \tau_k + \omega.$$

Therefore, the delay between the true convergence point and the detected convergence point is at most ω rounds. Since the per-round loss is bounded in $[0, 1]$, the regret incurred during this detection delay is bounded by $O(\omega)$ for each cluster.

Consequently,

$$\mathrm{Reg}_{\mathrm{det}} \le O(\sum_{k=1}^{K} \omega) = O(K\omega).$$

Combining all components,

$$\begin{aligned}\mathrm{Reg}(T) \le & \widetilde{O}\left(\sum_{k=1}^{K}\sqrt{T_k}\right) + \widetilde{O}\left((1+c)\sum_{k=1}^{K}\sqrt{\frac{T_k}{c}}\right) \\ & + O(cB^\star) + O(K\omega).\end{aligned}$$

Using Cauchy–Schwarz,

$$\sum_{k=1}^{K}\sqrt{T_k} \le \sqrt{KT}.$$

we obtain

$$\mathrm{Reg}(T) \le \widetilde{O}(\sqrt{KT}) + O(B^\star) + O(K\omega).$$

□

C Experimental Details

C.1 Implementation and Hyperparameters

Table 2 lists all settings used in the experiments. Cloud completions for every request are precomputed with Qwen3.5-Plus, so the cloud loss of every request is known and the frozen always-cloud policy is evaluated on the same request order as the adaptive methods. Always-local uses the untrained Qwen3-4B completion of each request. The cluster assigner seeds its score history with the first four requests, which are assigned to cluster 0; afterwards, the boundaries are the running $1/4$, $1/2$, and $3/4$ quantiles of all scores observed so far, computed by the local model at the time the request arrives. Running average curves in the main text start at $t = 200$.

C.2 Per-cluster Statistics on MATH

Table 3 reports the per-cluster operating point of the proposed algorithm. cluster 0 is served entirely by the fine-tuned local model, clusters 1 and 2 mostly so, and cluster 3 mostly by the cloud. The average number of completion tokens grows from 423 on cluster 0 to 1,523 on cluster 3, which again tracks problem difficulty, and only 7–13% of each cluster's rounds are spent on training.

C.3 Arm Selection on XSum

On clusters 0–2 the marginal improvement never falls below the detection threshold $\lambda = 0.1$, so train keeps roughly one third of the pulls through the end of the stream. Among the serving arms, the LCB still separates the clusters: cluster 0 favours the cloud (2,147 cloud vs. 583 local pulls), cluster 2 favours local (1,772 vs. 897), and cluster 1 is balanced. On cluster 3, 175 local pulls follow detection within 1,100 rounds at loss 1.009 against 0.929 for the cloud, after which the cloud serves the cluster.

Table 2: Hyperparameters of the experiments.

Setting	Value
Cloud model	Qwen3.5-Plus, thinking disabled
Local model	Qwen3-4B, LoRA ($r = 16$, $\alpha = 32$)
Fine-tuning	GRPO, 8 generations per prompt, cloud completion as teacher label
Decoding	temperature 1.0, top-$p = 0.95$, at most 2,048 new tokens
Token cost c_{tok}	2×10^{-4}
Adaptation cost c	0.1
MATH reward	exact match with true answer
XSum reward	token-level F1 against the reference answer
Number of clusters K	4 (running quartiles of the cluster score)
Convergence window ω	100
Consecutive detections	7
Detection threshold λ	0.3 (MATH), 0.1 (XSum)
Exploration coefficient	0.5 in all confidence bounds
Request batch size	16
GRPO step trigger	every 16 enqueued training examples
Random-p	$p = 0.3$

Table 3: Per-level statistics of the proposed algorithm on Hendrycks MATH (full loss).

Level	Avg. loss	Avg. tokens	Train %	Local %	Cloud %
0	0.122	423	12.9	87.1	0.0
1	0.213	650	12.1	81.8	6.1
2	0.384	967	7.3	62.0	30.7
3	0.793	1523	8.1	21.6	70.3

C.4 Clusters Induced by Different Criteria

Average token entropy yields balanced clusters whose mean official difficulty increases with the assigned cluster (2.86, 3.49, 3.83, 4.26), and the base-local versus cloud gap follows the same order. First-token entropy reverses the order (3.89, 3.44, 3.28, 3.22) and places only 224 requests in cluster 3: the first generated token carries little information about the difficulty of a multi-step solution, so the running quartiles collapse the hard tail into a nearly empty cluster and push most traffic into cluster 1. Prompt NLL is anti-monotone as well (3.98, 3.78, 3.41, 2.88): a likely prompt is not the same as an easy generation, because short, templated hard problems can be more predictable than long easy ones. Both alternatives trigger convergence early on their mixed clusters, after 611–1,701 rounds for first-token entropy and 997–3,008 rounds for prompt NLL, and spend only 1,002 and 1,572 train pulls against 2,362 for average token entropy. This is why their running average loss drops faster at first; average token entropy overtakes prompt NLL at round 6,946 and first-token entropy at round 7,781.

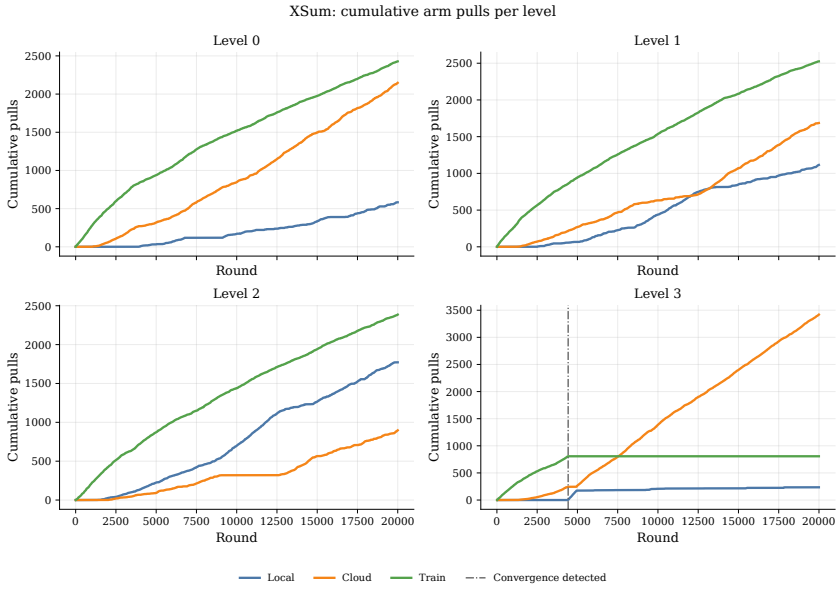


Fig. 4: Cumulative pulls of each arm on each XSum cluster. Clusters 0–2 never trigger convergence and keep all three arms active. Cluster 3 converges at round 4,424; its train curve freezes, a short burst of local pulls follows, and the cloud then serves the cluster.

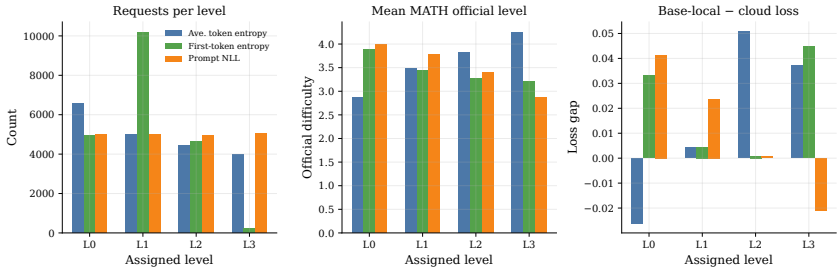


Fig. 5: Clusters induced by three clustering scores on MATH: number of requests, mean official MATH difficulty, and the gap between the frozen local model and the cloud. Average token entropy is the only score that is monotone in official difficulty and isolates a large hard cluster.