Prefill-Guided Trace Allocation for Sample-Efficient Test-Time Scaling

Zhi Yao[2,1] **Zhiqing Tang**[1,3,*] **Hanshuai Cui**[2,1]
Qianli Ma[3,1] **Fanshuai Meng**[3,1] **Weijia Jia**[1,3,4]

[1]Institute of AI and Future Networks, Beijing Normal University, Zhuhai, China
[2]School of Artificial Intelligence, Beijing Normal University, Beijing, China
[3]Faculty of Arts and Science, Beijing Normal University, Zhuhai, China
[4]Guangdong Key Lab of AI and Multi-Modal Data Processing,
Beijing Normal-Hong Kong Baptist University, Zhuhai, China

Abstract

Self-consistency is a widely adopted test-time scaling method that improves LLM reasoning by majority-voting over sampled traces. However, fixed-budget self-consistency applies the same trace count to every problem, even when the first greedy attempt already produces the correct answer, wasting substantial compute on resource-constrained hardware. Existing adaptive methods rely on post-decode agreement signals that become reliable only after complete traces have been generated, yet a prefill-stage oracle reveals that a large fraction of this cost can be eliminated before spending the full budget. We introduce the Hidden-State Adaptive Trace Scheduler (HATS), which predicts greedy-trace reliability from prefill hidden states before allocating additional sampled traces. HATS validates one greedy probe before routing easy problems to a single trace, borderline problems to a small vote, and hard problems to the full budget. A compute-optimal allocation analysis shows that extra samples should go where marginal accuracy gain is highest, explaining why non-uniform spending outperforms fixed budgets. On MATH-500 with DeepSeek-R1-Distill-Qwen-7B, HATS comes within 2.6 points of four-sample self-consistency accuracy while reducing samples by 48% and decode tokens by 40%, and with an eight-sample cap it matches eight-sample self-consistency accuracy with 52% fewer tokens. Without target labels, the same estimator keeps accuracy close to the four-sample target across eight models and eight mathematical benchmarks, and under continuous batching it answers 19% more problems per hour at matched concurrency. These results support prefill-guided scheduling as a practical compute-saving mechanism for test-time scaling.

1 Introduction

Large language models have demonstrated strong reasoning capabilities, particularly in mathematical problem solving. Test-time scaling (TTS) has emerged as the primary technique for improving their accuracy at inference time without additional training [1–3]. Among TTS methods, self-consistency (SC) [4] is one of the simplest and most widely adopted, returning the majority-vote answer from multiple sampled solutions. SC reliably improves accuracy, and four-sample voting is a common operating point. As distilled reasoners such as DeepSeek-R1-Distill-Qwen-7B [5] and Qwen3-8B [6] bring TTS to compact, resource-constrained hardware, the compute cost of multi-sample voting becomes a first-order concern.

*Corresponding author: Zhiqing Tang (zhiqingtang@bnu.edu.cn).

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

Table 1: Routing-gap summary around the four-sample SC target on MATH-500. "Online" denotes decisions made during generation without a precomputed trajectory pool.

Method group	Acc. target	Saves tokens	Online?	Saving
Post-decode tuned	✓	×	✓	-3.4 to -99.6%
Post-decode default	×	×	✓	-18.6 to -82.0%
AC replay	✓	✓	offline	38.0%
Oracle-PDE	✓	✓	oracle	48.1%
HATS	✓	✓	✓	40.2%

However, SC applies the same number of traces to every problem regardless of difficulty. For distilled reasoners whose first attempt is often already correct, this uniform budget is particularly wasteful. In our setting, greedy decoding alone solves the majority of problems, yet four-sample SC still allocates four traces to all of them. The resulting waste is not that easy problems need multiple traces, but that the scheduler discovers too late that extra traces were unnecessary. Existing adaptive methods try to reduce this cost. Post-decode stopping rules [7, 8] need complete traces before agreement evidence emerges, and self-rating approaches [9] prove unreliable in practice. These methods share a timing gap. The evidence needed for routing arrives only after the most expensive computation has already occurred. Therefore, prefill-guided trace allocation that routes problems before extra traces are decoded is needed to close this gap. Oracle analysis confirms that substantial headroom for such early routing exists (Table 1). Closing this timing gap requires addressing two challenges.

The first challenge is how to extract a reliable difficulty signal from prefill representations before any answer trace is decoded. The prefill hidden states of the reasoning model encode the problem but contain no sampled solution trace, making the difficulty ranking noisy. Unlike post-decode agreement, which directly observes answer consistency, a prefill signal must predict whether the greedy trajectory *will be* correct without seeing any of it. Concurrent activation-based methods [10–12] focus on binary gates, multi-model routing, or thinking-length budgets. Co-locating an external verifier such as a process-reward model [13, 14] is infeasible under the single-GPU memory target, so the estimator must rely on the reasoning model's own representations. It need not produce calibrated probabilities, but must rank greedy-correct problems above greedy-wrong ones well enough for a low-error routing threshold to be selected on validation data.

The second challenge is how to allocate trace budgets conservatively under asymmetric routing errors. A false easy route loses accuracy that additional sampling could have recovered, whereas a false hard route wastes compute but preserves correctness. This asymmetry rules out simple threshold routing and requires multi-stage guards that catch unreliable greedy traces before committing. Existing budget-estimation methods set per-problem sample counts [15, 16] or predict stopping points [17], but do not account for this asymmetry. Moreover, even with a reliable difficulty ranker, the optimal number of traces for borderline problems is not obvious. Uniform allocation is a natural default, but it is suboptimal when problems have different marginal gains from additional samples [18].

We propose the Hidden-State Adaptive Trace Scheduler (HATS), a prefill-guided scheduler built around a simple principle: predict whether extra traces are worth spending, not what the final answer is. HATS routes easy problems to a single greedy trace that passes its validity guards, borderline problems to a small vote, and hard problems to the full budget (Fig. 1). On MATH-500 with DeepSeek-R1-Distill-Qwen-7B, HATS comes within 2.6 points of the four-sample SC target while using 48.0% fewer samples and 40.2% fewer decode tokens, and with an eight-sample cap it matches the eight-sample target with 51.7% fewer tokens. Contribution (1) addresses the first challenge, contribution (2) addresses the second, contribution (3) provides the theoretical foundation, and contribution (4) gives scoped empirical validation. Our contributions are summarized as follows.

1. **Routing gap and prefill difficulty estimator.** We identify a timing gap in existing adaptive SC, where post-decode baselines either miss the accuracy target or exceed the token budget. To address this, we propose PDE+, a lightweight estimator that ranks greedy-trace reliability from the reasoning model's own prefill hidden states.
2. **Conservative multi-stage scheduler.** We design HATS, which combines PDE+ with a checked greedy probe, parser and log-probability guards, and adaptive majority voting. HATS routes problems conservatively under asymmetric errors and requires no external verifier or additional language model.

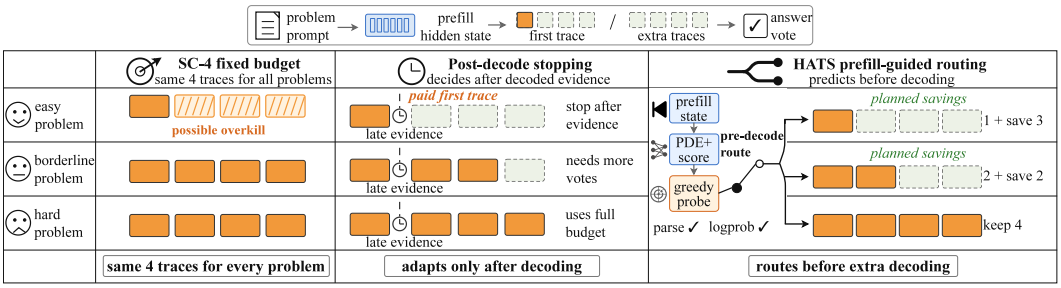


Figure 1: Overview of HATS allocation around a four-sample SC target. Prefill routing sends easy, borderline, and hard problems to one checked trace, a small vote, and the full budget, respectively.

3. **Allocation analysis and routing regret.** Using a water-filling analysis, we show that uniform sampling is suboptimal under a fixed compute budget. A routing-regret bound then links realized saving to prefill ranking quality.
4. **Empirical characterization.** On DeepSeek-R1-Distill-Qwen-7B we evaluate calibration robustness, threshold sensitivity, ablations, and faithful reproductions of the closest prefill-routing baselines. The same fixed protocol, with no target labels, extends to eight models from 7B to 32B and eight mathematical benchmarks, and under continuous batching the token saving surfaces as higher throughput and lower GPU time per correct answer.

2 Related Work

Test-time scaling encompasses tree search [19], action-based reasoning [20], iterative refinement [21, 22], and verification-based methods including best-of-N reranking [13] and process-reward models [14, 23]. HATS targets the self-consistency branch.

Adaptive self-consistency. Fixed-budget SC allocates the same trace budget to easy and hard instances. Adaptive SC methods reduce this cost using post-generation answer agreement [7, 8, 24], confidence or reasoning-path signals [25–28], and budget-estimation or pruning heuristics [15, 29, 16, 17]. Unlike these post-decoding policies, HATS ranks greedy-trace reliability from prefill states before generating additional traces, using a single fixed reasoner rather than a model cascade. Implementation details for the DSC/RPC-style rows are given in Appendix C.

Activation-based routing. Internal states have been used to route computation at test time. ACTSC [10] uses activation features to make a binary pre-sampling decision (apply SC or greedy alone), [11] use prefill activations for multi-model routing, Adaptive Thinking [12] predicts self-consistency from prefill states to set a thinking-token budget, and ReProbe [30] probes mid-generation states for adaptive scaling. HATS differs in its prediction target, the reliability of the specific greedy trace rather than dataset difficulty or sampling agreement, and in performing graded allocation (checked-greedy, small-vote, or full-SC) with conservative post-decode guards under a fixed SC-4 accuracy reference. Section 6.4 reproduces ACTSC and LLM Router on our pools and separates scorer quality from allocation policy.

Compute allocation. Compute-optimal test-time scaling shows that non-uniform sample allocation can outperform uniform spending under oracle or search-based policies [18]. Cost-aware routing systems such as FrugalGPT [31] and efficient serving engines [32–34] address complementary aspects of inference cost. HATS learns a prefill-time ranking signal for trace allocation under a fixed accuracy target on a single model.

3 Motivation

3.1 Setup and fixed-budget slack

We study whether an adaptive scheduler can match the four-sample SC (SC-4) accuracy target while spending fewer decoded traces. We isolate this routing problem on DeepSeek-R1-Distill-Qwen-7B,

MATH-500, a shared answer parser, and one 24 GB GPU. SC-4 serves as the fixed accuracy–compute target, using one greedy trace plus three sampled traces on every problem. SC-8 doubles this fixed trace budget. The scheduler may vary the number of traces allocated to each problem, but is evaluated by whether it preserves four-sample self-consistency accuracy. Wall-clock timing depends on serving orchestration and is reported separately in Section 6.5.

Fixed-budget SC leaves substantial slack. Greedy decoding solves 71.4% of MATH-500; SC-4 adds 11.8 points, while SC-8 adds less than 1 point. Compute varies widely, with SC-4 token counts ranging from about $2{,}500$ to the $16{,}384$ cap and about 70% of problems receiving unanimous SC-4 votes. Uniform sampling spends extra traces on examples whose first trajectory was already sufficient.

3.2 Post-decode plateau and prefill-stage headroom

Table 1 summarizes the routing gap as three constraints: matching the SC-4 accuracy target, saving decode tokens, and remaining online. Completed end-to-end post-decode baselines in our vLLM stack fail at least one of these constraints: tuned variants recover or exceed the SC-4 accuracy target but remain over budget, while default operating points miss the target and still spend more tokens than SC-4. Stopping after observing decoded answers is therefore not enough for an online equal-accuracy token saver.

The same summary also shows why the gap is worth pursuing. A trace-driven AdaptiveConsistency replay reaches the SC-4 target with 38.0% token saving, but only after the fixed SC-4 trajectory pool has already been generated. It is therefore an offline diagnostic rather than an online baseline. Conversely, Oracle-PDE, a prefill-stage router that knows greedy correctness, matches SC-4 with 48.1% fewer tokens. Together, the two points show that useful routing signal exists and that its timing matters: decoded agreement becomes reliable only after complete traces are paid for, whereas a prefill-stage signal could avoid them.

4 The HATS framework

4.1 Overview

HATS is an adaptive scheduler that runs on top of vLLM [32]. It decomposes adaptive self-consistency into two decisions made before most extra decoding cost is incurred: estimating whether the first greedy trace is likely to be reliable, and deciding how many additional traces to allocate under the SC-4 cap. The first decision is supported by a learned prefill-only difficulty estimator, PDE+, which scores each problem's difficulty before decoding. The second is implemented by the Multi-Stage Adaptive Probe (MSAP; Algorithm 1), which combines this score with a checked greedy probe and adaptive voting to choose among three routing paths. The checked-greedy path commits the greedy trace once it passes the validity guards, the small-vote path samples until two traces agree, and the full-budget path samples until three agree or the cap is reached. Following the allocation view of Section 5, extra samples should go where their marginal gain is high rather than uniformly across problems, so HATS does not require calibrated per-problem accuracy curves or a prediction of the final answer. It only needs a prefill-time ranking signal that separates problems likely to be solved by greedy decoding from those that should keep access to the SC budget. PDE+ supplies this ranking. The greedy probe, parse check, log-probability guard, and vote-margin rules then make the routing conservative when the first trace is unreliable.

Notation. We use $K_{\max} = 4$ for the fixed SC-4 cap, y_1 for the first greedy trajectory, and $d \in [0, 1]$ for the PDE+ difficulty score, where larger values indicate harder problems. We write $\bar{\ell}$ for the token-mean log-probability of the greedy trace. The thresholds τ_{low} and τ_{lp} control the easy route and log-probability guard, respectively, and k_{stop} denotes the adaptive vote threshold.

4.2 Prefill difficulty estimation

PDE+ is a lightweight prefill estimator used to rank problems by greedy-trace reliability. Given a problem prompt, it outputs a scalar difficulty score $d \in [0, 1]$ before any answer trace is decoded, where larger values indicate higher estimated risk that the first greedy trace will be wrong. The "+" marks the version deployed in HATS, whose output is a routing score rather than a calibrated probability.

The deployed estimator reads the reasoning model's final-layer hidden states averaged over the last five prompt tokens and passes them to a two-layer multilayer perceptron (MLP) with hidden sizes 512 and 128, trained with binary cross-entropy on greedy-correctness labels from a single greedy pass. Prompt-level text features appear only in the ablation of Table 16. Under strict out-of-fold evaluation the scorer reaches an area under the ROC curve (AUC) of 0.73 for greedy-wrong ranking on MATH-500, and the AUC saturates at roughly 100 to 200 labels per fold (Appendix D). Table 15 and Figure 2 report its ranking quality and the resulting routing.

4.3 Runtime scheduling

Algorithm 1 gives the inference-time procedure. HATS first decodes a greedy trajectory y_1 at temperature 0, providing the cheapest candidate answer and runtime checks unavailable at prefill time. It then computes the token-mean log-probability $\bar{\ell}$ and parser-validity flag v_1.

HATS commits to the greedy answer only when the trace passes the validity guards, a parseable final answer that was not truncated at the generation cap, and either the prefill estimate marks the problem easy ($d < \tau_{\text{low}}$) or the problem is borderline ($d < 0.5$) and the trace is confident ($\bar{\ell} > \tau_{\text{lp}}$). Otherwise, it samples additional traces at temperature 0.7, updating the majority vote after each trace. The stopping threshold is $k_{\text{stop}} = 2$ for borderline problems with $d < 0.5$ and $k_{\text{stop}} = 3$ otherwise. Sampling stops once the leading answer reaches this threshold or the SC-4 cap $K_{\max} = 4$.

Algorithm 1 MSAP: the HATS scheduler

Require: problem x, PDE+ scorer, thresholds $\tau_{\text{low}}, \tau_{\text{lp}}$, sample cap $K_{\max}$

$d \leftarrow \text{PDE}^{+}(x)$ ▷ prefill difficulty score
$y_1 \leftarrow \text{Greedy}(x)$
$\bar{\ell} \leftarrow \text{TokenMeanLogProb}(y_1)$
$v_1 \leftarrow [y_1 \text{ parseable and not truncated}]$
if v_1 **and** $d < \tau_{\text{low}}$ **then**
  return y_1 ▷ easy: guarded greedy commit
else if v_1 **and** $d < 0.5$ **and** $\bar{\ell} > \tau_{\text{lp}}$ **then**
  return y_1 ▷ borderline but confident
end if
$\mathcal{Y} \leftarrow \{y_1\}$
$k_{\text{stop}} \leftarrow 2$ **if** $d < 0.5$ **else** 3
for $k = 2, \ldots, K_{\max}$ **do**
  $y_k \leftarrow \text{Sample}(x, \text{temp} = 0.7)$
  $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_k\}$
  if $\text{TopVoteCount}(\mathcal{Y}) \geq k_{\text{stop}}$ **then**
    break
  end if
end for
return $\text{MajVote}(\mathcal{Y})$

4.4 Budget-aware threshold calibration

HATS uses a small set of routing thresholds: the easy-route threshold τ_{low}, log-probability guard τ_{lp}, adaptive-voting thresholds, and sample cap $K_{\max}$. They are selected once and then fixed. Let $S_q(\theta)$ denote the trace count for calibration problem q, and $M_q(\theta)$ its peak KV-cache footprint under thresholds θ. Given a budget $\mathcal{B} = (C_{\text{s}}, M_{\text{KV}}, \delta)$, we choose thresholds by maximizing calibration accuracy subject to

$$\mathbb{E}_q\big[S_q(\theta)\big] \leq C_{\text{s}}, \qquad \Pr_q\big[M_q(\theta) \leq M_{\text{KV}}\big] \geq 1 - \delta. \tag{1}$$

In the SC-4-target setting, we keep $K_{\max} = 4$, so no problem receives more traces than the fixed-budget reference. A 200-trial random search over replayed traces selects $\tau_{\text{low}} = 0.30$, $\tau_{\text{lp}} = -0.40$, and the adaptive-voting rule in Algorithm 1. Each trial replays the cached traces on a CPU, so the search completes in minutes. The search uses out-of-fold probe scores on the same 500-problem pool that we report. The selected thresholds sit on a broad plateau, where all 180 settings of a $\tau_{\text{low}} \times \tau_{\text{lp}}$ grid reach 81.8–82.0% accuracy at 37.8–42.0% saving. A nested protocol, which selects the thresholds for each fold on the remaining folds only, gives 82.0% accuracy at 42.6–42.7% saving over three seeds (Appendix D). Transfer evaluation keeps both the MATH-trained PDE+ and these thresholds fixed and uses no target labels.

5 Allocation Analysis

5.1 Budgeted allocation model

We formalize adaptive trace allocation as budgeted sample allocation. For N test problems indexed by $i \in [N]$, let $\text{Acc}_i(n)$ denote the expected majority-vote accuracy using $n \in \{1, \ldots, K\}$ samples,

where K is the per-problem cap. A scheduler chooses integer budgets n_i under a global compute constraint $\sum_i n_i \le B$.

This abstraction captures the allocation question in fixed-budget self-consistency. If the problem-specific accuracy–sample curves were known, an oracle scheduler would spend extra samples where marginal value is highest, rather than assigning every problem the same number of traces. HATS approximates this principle by using PDE+ to rank which problems likely need more than the greedy trace. The model abstracts away token-length variation, parser failures, and sample dependence. The experiments use real ordered vLLM sample streams and measure those effects directly.

We assume diminishing marginal utility. Let $g_i(n) = \mathrm{Acc}_i(n) - \mathrm{Acc}_i(n-1)$ be the marginal gain of the n-th sample on problem i. We assume $g_i(n) \ge 0$ and $g_i(n+1) \le g_i(n)$ for $n = 1, \ldots, K-1$. This is a stylized model of the slack described in Section 3.1: once a problem is likely to be solved, additional traces have lower expected value than traces spent on unresolved or borderline problems.

5.2 Compute-optimal allocation

The idealized allocation problem is

$$\max_{\{n_i\}} \sum_{i=1}^{N} \mathrm{Acc}_i(n_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} n_i \le B, \quad 1 \le n_i \le K, \quad n_i \in \mathbb{Z}. \tag{2}$$

Starting from one mandatory sample per problem, the optimal policy spends each additional sample where its feasible marginal gain is largest.

Proposition 1 (Compute-optimal sample allocation)**.** Under diminishing marginal utility, an optimal integer allocation for Eq. (2) is obtained by selecting the $B - N$ largest feasible marginal increments $\{g_i(2), \ldots, g_i(K)\}$ subject to prefix feasibility (selecting the m-th increment for problem i requires selecting all earlier increments $2, \ldots, m-1$). Equivalently, there exists a threshold λ such that every selected increment has marginal gain at least λ, and every unselected feasible next increment has marginal gain at most λ.

This result explains why uniform SC can be wasteful. When problems have different marginal gains from additional traces, equal sampling is optimal only in the special case where those marginal gains align. The allocation rule therefore depends primarily on ranking problems by the value of extra sampling, motivating PDE+ as a ranker for greedy-trace reliability.

5.3 Iso-accuracy routing regret

We next relate the allocation view to a learned prefill router. Let $\pi^\star$ be an oracle prefill router that knows whether greedy decoding is correct, and let π_θ be a threshold router using PDE+ scores. The router commits to the greedy trace when the score is below threshold τ and escalates otherwise. Let C^τ and C^{oracle} denote the per-problem sample cost under the learned and oracle routers, respectively.

Proposition 2 (Iso-accuracy routing regret)**.** Let ρ be the greedy-correct rate. Let $\mathrm{FPR}(\tau)$ be the false-commit rate and $\mathrm{FNR}(\tau)$ the false-escalation rate. Under iso-accuracy accounting, where false commits are charged by the extra SC budget that would have avoided the error, the expected per-problem routing regret relative to the oracle is

$$\mathbb{E}[C^\tau - C^{\mathrm{oracle}}] \le (K-1)\Big((1-\rho)\,\mathrm{FPR}(\tau) + \rho\,\mathrm{FNR}(\tau)\Big) + \mathcal{O}(N^{-1/2}). \tag{3}$$

The bound depends on the estimator only through the weighted routing error. A threshold router therefore needs the score to rank greedy-correct problems ahead of greedy-wrong ones, which is what AUC measures. Appendices A and B give the proofs, and Table 15 reports the empirical connection between estimator quality and realized saving.

6 Experiments

6.1 Setup

Models. We evaluate compact reasoning models because adaptive trace allocation is most useful when the base model is fixed and the test-time budget is constrained. The main reasoner is DeepSeek-

Table 2: MATH-500 accuracy–compute results. Token saving is relative to the fixed-budget reference within each block. Replay and oracle rows are diagnostic references rather than online baselines.

Method	Stage	Acc. (%)	Avg. samples	Token saving vs. ref.	Outcome
SC-4 target					
SC-4 (fixed budget)	none	83.20	4.00	0.0%	target
AC [8]	post-decode	75.80	5.81	−82.0%	miss + over
AC (τ=0.70 tuned)	post-decode	85.40	3.52	−3.4%	over budget
ESC (W=2) [7]	post-decode	86.20	4.81	−57.0%	over budget
TVS (k=2)	post-decode	75.20	4.26	−43.5%	miss + over
DSC-style [15]	post-decode	87.80	6.58	−99.6%	over budget
RPC-style [29]	post-decode	74.80	3.82	−18.6%	miss + over
DynaThink [9]	self-rating	71.40	1.00	–	miss target
HATS (Ours)	prefill+probe	**80.60**	**2.08**	**40.2%**	target + save
Oracle-PDE (UB)	oracle	83.20	2.07	48.1%	upper bound
SC-8 target extension					
SC-8 (fixed budget)	none	83.40	8.00	0.0%	target
HATS-K8 (Ours)	prefill+probe	**83.60**	**3.30**	**51.7%**	target + save

R1-Distill-Qwen-7B [5] in BF16. The same protocol is then run on seven further models, DeepSeek-R1-Distill-Qwen-14B and 32B, DeepSeek-R1-Distill-Llama-8B [5], Qwen3-8B [6], Qwen3.5-9B, Qwen3.6-27B, and Gemma-3-12B-IT [35], each served in its native prompt format with matched generation caps for all arms. All generations are produced with vLLM [32].

Datasets. We evaluate on MATH-500 (full, 500 problems), a fixed 500-problem subset of GSM8K test, and an AIME 2022–2024 suite (90 problems: AIME I and II from each year), and additionally run the 7B model zero-shot on SVAMP (300), GSM-Hard (300), MinervaMath (272), AMC23 (40), and AIME 2025 (30).

Decoding. Sampled traces use temperature 0.7 and top-$p = 0.95$. We cap generation at 4,096 tokens on MATH-500 and GSM8K, and at 24,576 tokens on the AIME 2022–2024 suite because its solutions often require substantially longer reasoning chains. Context windows are set to 5,120 and 25,600 tokens, respectively. vLLM memory settings are given in Appendix D.

Metrics. We report answer accuracy, average decoded tokens per problem, and average number of generated traces. Accuracy uses a shared parser based on `math_verify` extraction and SymPy equivalence checks. Serving throughput, GPU time per correct answer, and latency under continuous batching are measured in Section 6.5.

Baselines. Fixed SC-K uses one temperature-zero greedy trace followed by $K - 1$ stochastic traces at temperature 0.7. The SC-K and HATS arms share the problems, prompts, and the deterministic greedy trace, while each arm samples its own stochastic traces. HATS saves compute by deciding how many of these traces to allocate. We also include post-decode adaptive stopping rules: ESC [7] ($W = 2$), AdaptiveConsistency (AC) [8] (threshold 0.90), and TVS ($k = 2$), which is the vote-count special case of ESC discussed in Section 2. In addition, we evaluate DynaThink [9] and same-stack online adaptations inspired by DSC [15] and RPC [29], with implementation details in Appendix C. We also reproduce ACTSC and LLM Router on our cached MATH-500 traces and adapt Adaptive Thinking's length-budget executor to our stack (Appendix C).

Baseline evaluation protocol. Table 2 reports end-to-end runs using the same model, parser, and vLLM setup as HATS: online rows decide when to stop in this shared stack, replay rows select from an already generated SC-4 pool and are diagnostic only, and DSC/RPC-style rows are same-stack proxy adaptations (Appendix C).

6.2 Main result on MATH-500

Table 2 reports the main MATH-500 comparison. Under the SC-4 target, HATS reaches 80.60% accuracy, within 2.6 points of SC-4, while reducing decoded tokens by 40.2% and samples by 48.0%. PDE+ routes most easy problems to the checked-greedy path, avoiding unnecessary traces while retaining the voting budget for hard problems. Paired bootstrap intervals exclude zero for the accuracy difference and for both savings, while the strict out-of-fold run narrows the accuracy gap to 1.6

points with an interval that includes zero (Appendix D). HATS recovers 83.6% of the Oracle-PDE upper-bound token saving ($40.2/48.1$).

The SC-8 block provides a higher-budget extension. After recalibration around the SC-8 reference, HATS-K8 reaches 83.60% accuracy while saving 51.7% tokens relative to SC-8. With a larger cap, more per-problem traces are available to eliminate, amplifying the saving from adaptive allocation. The 0.2-point accuracy difference is within noise (McNemar $p = 1.00$, Appendix D).

Table 3 summarizes calibration checks for the SC-4 setting. The main calibrated, strict out-of-fold, and external-calibration protocols all land within 2.6 points of SC-4. The strict out-of-fold run reaches 81.60% accuracy with 39.9% token saving, and external calibration reaches 81.20% with 25.7%. These checks suggest that the saving is not driven by fold overlap in the PDE+ calibration, although external calibration is more conservative because a PDE+ estimator trained on a different distribution produces less confident scores and commits fewer problems to the greedy path. Appendix D gives the corresponding protocols. Bootstrap intervals capture problem-level uncertainty rather than seed-to-seed variance.

Table 3: Calibration robustness on MATH-500. Strict OOF gives each problem a prefill score from a model trained without that problem's fold; external calibration uses no MATH-500 examples for estimator training or threshold selection. Rows are HATS calibration protocols, not separate methods.

Protocol	Acc. (%)	Tokens	Saving
Calibrated	80.60	5,761	40.2%
Strict OOF	81.60	5,786	39.9%
External	81.20	7,160	25.7%

The post-decode baselines can reach high accuracy, but their stopping evidence arrives only after decoded samples exist, and all high-accuracy completed runs remain over budget. The headline claim is based on online end-to-end HATS runs.

HATS commits 58.2% of problems after one checked greedy trace, routes 19.4% through a small-vote path, and sends only 22.4% to the full SC-4 budget (Appendix D).

6.3 Zero-shot transfer

Table 4 applies each model's MATH-trained PDE+ to GSM8K and the AIME 2022–2024 suite with the thresholds fixed and no target labels. On DeepSeek-7B the accuracy gap stays within 2.8 pp while the allocated budget grows with difficulty, from 1.80 samples per problem on GSM8K to 3.00 on AIME. Qwen3-8B shows the same pattern. Tables 8 and 9 in Appendix D extend the protocol to seven further models from three families and five further datasets. Across them the worst accuracy loss is 3.3 pp, saving ranges from 12.6% to 76.3%, and the allocated budget tracks dataset difficulty, rising from 1.56 to 3.38 samples per problem (Spearman $\rho = 0.88$ against the SC-4 error rate). MATH saving also grows with base-model accuracy ($\rho = 0.88$ across the eight models), since stronger greedy passes make more problems safe to commit.

Table 4: Zero-shot transfer relative to same-pool SC-4. Each model's PDE+ is trained once on its own MATH-500 and applied with unchanged thresholds and no target labels. MATH rows are in-domain.

Model / dataset	ΔAcc.	Token saving
DeepSeek-7B / MATH	−2.6 pp	40.2%
DeepSeek-7B / GSM8K	−2.8 pp	53.3%
DeepSeek-7B / AIME	−2.2 pp	15.5%
Qwen3-8B / MATH	−1.8 pp	44.6%
Qwen3-8B / GSM8K	−1.0 pp	73.2%
Qwen3-8B / AIME	+0.0 pp	17.8%

6.4 Ablation

Table 6 isolates the two mechanism choices most tied to the main claim. Removing the checked-greedy commit, so that every problem receives at least two traces, raises accuracy to 82.4% but costs 28% more tokens, because majority voting recovers errors that greedy alone would miss. The greedy commit therefore turns the scheduler into a token-saving method. Replacing PDE+ with a constant score saves 10% more tokens than HATS but lowers accuracy by 1.6 points, showing that the prefill ranking is what preserves accuracy while saving tokens. Figure 2 adds bucket-level routing diagnostics: strict out-of-fold MATH-500 scores order problems by greedy reliability and rescue value, while GSM8K and AIME show the same diagnostic under target-calibrated PDE+ estimators.

Table 5: Prefill-routing baselines and policy decomposition, replayed on the cached MATH-500 traces (SC-4 reference 83.2% and 9,634 tokens). The HATS replay row tracks the reported run of Table 2. Scorer AUC is out-of-fold against greedy correctness, the quantity the commit decision acts on.

Method (faithful configuration)	Scorer AUC	Acc. (%)	vs. SC-4	Token saving
ACTSC [10]	0.636	77.8	−5.4 pp	30.0%
LLM Router [11], own binary gate at iso-accuracy	0.742	82.4	−0.8 pp	10.3%
LLM Router scorer inside the HATS policy	0.742	81.6	−1.6 pp	43.9%
Binary gate + guards, PDE+ scores	0.727	81.8	−1.4 pp	38.1%
HATS with random scores (5 seeds)	–	82.0	−1.2 pp	30.5%
HATS (deployed thresholds, replay)	**0.727**	**81.8**	**−1.4 pp**	**41.4%**

Table 6: Routing ablations: checked-greedy commit saves compute, and PDE+ preserves accuracy.

Variant	Acc. (%)	Tokens	Samples	Outcome
No checked-greedy commit	82.40	7,351	2.82	accuracy lift, costlier
Constant PDE+ score	79.00	5,196	1.90	misses target
HATS (Ours)	**80.60**	**5,761**	**2.08**	target + save

Table 5 compares HATS with faithful reproductions of the closest prefill-routing methods and separates scorer from policy. ACTSC's probe predicts dataset difficulty rather than this model's greedy reliability and loses the most accuracy. The LLM Router scorer ranks greedy correctness slightly better than PDE+, yet its binary gate must send about 80% of problems to full SC-4 to hold accuracy and saves a quarter of what HATS saves. Moving the same scorer into the HATS policy recovers the full saving, so the gap is due to the allocation policy. The guards keep accuracy robust and the ranking buys the saving. With random scores accuracy holds but saving drops by a quarter, and a guarded binary gate on PDE+ scores saves less than graded escalation. An Adaptive Thinking-style length-budget executor reaches only 58.4% because 42% of its traces truncate, and any single-trace length policy is capped by greedy accuracy, 71.4%.

6.5 Serving-level measurement

In a single-stream prototype HATS is slower than SC-4 despite emitting fewer tokens, because the GPU idles between its greedy probe, guard checks, and small voting calls (Table 17 in Appendix D). Under continuous batching, the standard serving path, other problems' traffic fills those idle bubbles. Table 7 runs both methods on a stock vLLM server with prefix caching, closed-loop over MATH-500 with problem-level concurrency and identical prompts, caps, and hardware. At matched concurrency HATS completes 19.4% more problems per hour at 15.0% lower GPU time per correct answer. Because HATS holds fewer KV-cache slots per problem, the same GPU also serves it at concurrency 64, where the token saving surfaces almost fully as throughput, while SC-4 degrades beyond 32. Neither accuracy difference is statistically significant (McNemar $p = 0.45$ and 0.34), and all configurations run at about 100% GPU utilization. The same pattern holds on a datacenter A800-80GB GPU, where SC-4 peaks at concurrency 128 while HATS keeps scaling to 384 (Appendix D). Median latency favors HATS, since 58% of problems finish after one greedy trace, while P95 and P99 favor SC-4 because escalated problems pay two rounds. HATS therefore improves throughput and cost per correct answer but not tail latency.

7 Discussion and Limitations

Scope and limitations. HATS saves the most where greedy decoding is reliable and spends more budget, for a smaller saving, on harder distributions such as AIME and MinervaMath. On GSM8K the transferred score ranks greedy correctness near chance (AUC 0.54), the commits are decided mainly by the guards, and a 2.8-point loss remains across thresholds and vote depths. Zero-shot transfer therefore does not always preserve the SC-4 target. On MATH-500 itself HATS stays 2.6 points below SC-4 at $K_{\max} = 4$ and matches SC-8 at $K_{\max} = 8$. The guards rely on parseable final answers and SymPy equivalence, which limits the evaluation to mathematical reasoning, and each model needs about 500 greedy-correctness labels to train its PDE+ estimator. The throughput and

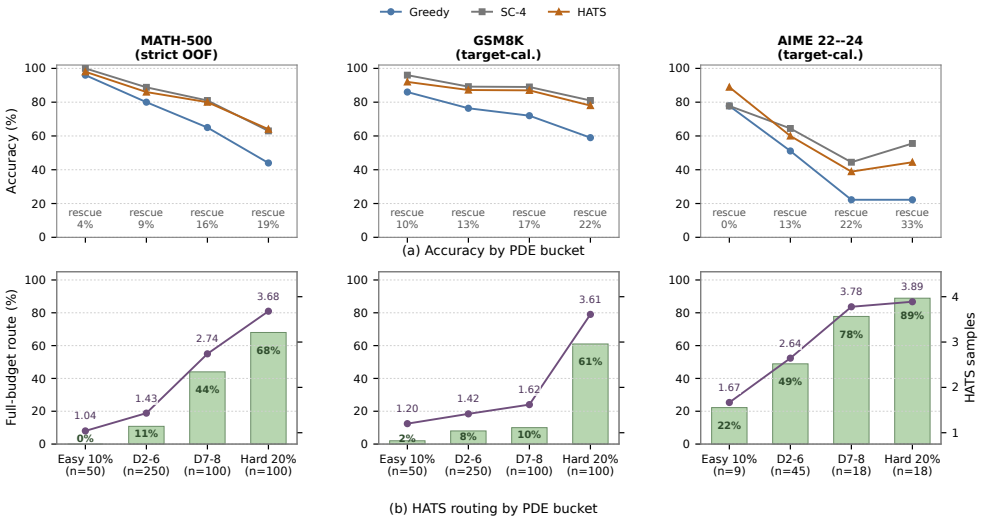


Figure 2: PDE+ bucket diagnostics across datasets (within-dataset quantiles by predicted difficulty). MATH-500 uses strict out-of-fold scores. GSM8K and AIME use target-calibrated probes.

Table 7: Serving under continuous batching on one RTX 3090. SC-4 is at its best concurrency (32), and HATS, holding fewer KV slots per problem, also runs at 64.

Metric	SC-4, conc. 32	HATS, conc. 32	HATS, conc. 64
Problems per hour	573.5	684.7 (+19.4%)	950.5 (+65.7%)
GPU-seconds per correct answer	7.60	6.46 (−15.0%)	4.66 (−38.7%)
Accuracy	82.6%	81.4%	81.2%
Median / P95 / P99 latency (s)	198 / 301 / 322	70 / 445 / 448	95 / 626 / 664

cost gains require continuous batching, and tail latency favors the fixed budget because escalated problems pay two rounds.

Broader impact. Reducing inference tokens lowers energy use and serving cost. Because routing errors can silently degrade accuracy on individual problems, adaptive allocation should be validated on the target distribution before high-stakes use.

Why compact models on constrained hardware. We develop HATS on 7–9B models on single-GPU hardware, where token savings matter most. A single 24 GB GPU has limited KV-cache headroom and cannot co-locate a process-reward verifier, so the estimator reuses the reasoner's own prefill states. Compact reasoners already solve most problems with one greedy trace (71.4% on MATH-500), which supplies the large easy-problem population that adaptive allocation needs. The saving also carries over to a datacenter A800 under continuous batching.

8 Conclusion

HATS reframes adaptive self-consistency as a prefill-guided trace-allocation problem around a fixed SC target. A prefill estimator ranks problems by the reliability of their greedy trace, and a guarded scheduler commits easy problems to that trace while sending the rest to a small vote or the full budget. On MATH-500 HATS comes within 2.6 points of the SC-4 accuracy target while reducing decode tokens by 40.2% and samples by 48.0%, and it matches the SC-8 target with 51.7% fewer tokens. Strict out-of-fold evaluation preserves the saving. Without target labels, the estimator keeps accuracy within a bounded envelope across eight models and eight mathematical benchmarks, and under continuous batching the saving surfaces as throughput, with tail latency favoring the fixed budget. Future work includes larger SC targets, non-mathematical tasks, and latency-aware scheduling.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grants 62272050 and 62302048, in part by the Guangdong Key Lab of AI and Multi-modal Data Processing, Beijing Normal-Hong Kong Baptist University, Zhuhai, under Grant 2023-2024 sponsored through the Guangdong Provincial Department of Education, in part by the Institute of Artificial Intelligence and Future Networks (BNU-Zhuhai) and the Engineering Center of AI and Future Education, Guangdong Provincial Department of Science and Technology, China, in part by the Zhuhai Science-Tech Innovation Bureau under Grant 2320004002772, and in part by the Interdisciplinary Intelligence SuperComputer Center of Beijing Normal University (Zhuhai). The authors declare no competing interests.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[2] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[3] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[4] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Daya Guo et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, Sept. 2025. doi: 10.1038/s41586-025-09422-z.

[6] An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[7] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396. Association for Computational Linguistics, 2023.

[9] Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. DynaThink: Fast or slow? a dynamic decision-making framework for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14686–14695. Association for Computational Linguistics, 2024.

[10] Taewoong Yoon, Geunyeong Jeong, Geon Park, Sihyeong Yeom, and Harksoo Kim. Breaking the pre-sampling barrier: Activation-informed difficulty-aware self-consistency. *arXiv preprint arXiv:2602.09438*, 2026.

[11] Tanay Varshney, Annie Surla, Michelle Xu, Gomathy Venkata Krishnan, Maximilian Jeblick, David Austin, Neal Vaidya, and Davide Onofrio. LLM router: Rethinking routing with prefill activations. *arXiv preprint arXiv:2603.20895*, 2026.

[12] Pingzhi Li, Bairu Hou, Yun Zhu, Yihao Feng, Ke Ye, Tao Lei, Zhifeng Chen, Tianlong Chen, and Xianzhi Du. Adaptive thinking: Large language models know when to think in latent space. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=2i6Rp0gCq6.

[13] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 9426–9439. Association for Computational Linguistics, 2024.

[15] Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6919–6932. Association for Computational Linguistics, 2025.

[16] Austin Feng, Marius Alonso, Ambroise Odonnat, Vasilii Feofanov, and Ievgen Redko. Optimal self-consistency for efficient reasoning with large language models. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

[17] Shiyu Ji, Yixuan Wang, Yijun Liu, Qingfu Zhu, and Wanxiang Che. Seer self-consistency: Advance budget estimation for adaptive test-time scaling. *arXiv preprint arXiv:2511.09345*, 2026.

[18] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[19] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2023.

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[21] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc., 2023.

[22] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023.

[23] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[24] Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning. *arXiv preprint arXiv:2510.10103*, 2025.

[25] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111. Association for Computational Linguistics, 2025.

[26] Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026.

[27] Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3613–3635. Association for Computational Linguistics, 2025.

[28] Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. Slim-SC: Thought pruning for efficient scaling with self-consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34500–34517. Association for Computational Linguistics, 2025.

[29] Zhi Zhou, Yuhao Tan, Zenan Li, Yuan Yao, Lan-Zhe Guo, Yu-Feng Li, and Xiaoxing Ma. A theoretical study on bridging internal probability and self-consistency for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pages 87380–87413. Curran Associates, Inc., 2025.

[30] Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. Reprobe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models. *arXiv preprint arXiv:2511.06209*, 2026.

[31] 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.

[32] 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*, page 611–626. Association for Computing Machinery, 2023.

[33] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*, 2024.

[34] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023.

[35] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

A Proof of Proposition 1

Proof of Proposition 1. Start from the mandatory one sample per problem. Any feasible allocation can be represented by a set of additional increments

$$\mathcal{S}(\mathbf{n}) = \{(i, m) : 2 \le m \le n_i\},$$

where increment (i, m) contributes marginal value

$$g_i(m) = \mathrm{Acc}_i(m) - \mathrm{Acc}_i(m-1).$$

Prefix feasibility means that selecting (i, m) also requires selecting $(i, 2), \ldots, (i, m-1)$.

Consider any feasible allocation $\mathbf{n}$. If there exist problems i and j with $n_i > 1$, $n_j < K$, and

$$g_i(n_i) < g_j(n_j + 1),$$

then moving one sample from i to j preserves the total budget and prefix feasibility and raises the objective by $g_j(n_j+1) - g_i(n_i) > 0$, contradicting optimality. Therefore, at an optimum, no selected removable increment has lower value than any feasible next unselected increment.

Conversely, suppose this no-improving-exchange condition holds. Because the objective is separable and each marginal sequence is non-increasing, any feasible reallocation can be decomposed into one-sample transfers that remove a selected last increment from one problem and add a feasible

next increment to another. If none of these transfers improves the objective, no feasible reallocation can improve it. Hence the allocation is obtained by selecting the $B - N$ largest feasible marginal increments subject to prefix feasibility. Equivalently, there exists a threshold λ such that selected increments have marginal value at least λ, while feasible next unselected increments have marginal value at most λ. □

Implication. Uniform self-consistency is optimal only when its selected marginal increments already satisfy this condition. Whenever a low-value selected increment under uniform allocation can be exchanged for a higher-value feasible increment on another problem, adaptive allocation has headroom.

B Proof of Proposition 2

Proof of Proposition 2. We analyze a binary prefill router. Let $y_i = 1$ denote that greedy decoding is correct on problem i, and let $y_i = 0$ otherwise. The oracle commits after one sample when $y_i = 1$ and escalates to K samples when $y_i = 0$. A learned threshold router commits when the PDE+ score is below threshold τ and escalates otherwise. Let C^{τ} and C^{oracle} be the iso-accuracy accounting costs of the learned router and oracle. We charge each router-oracle disagreement by at most $K - 1$ samples, counting false escalations as wasted samples and false commits as the SC budget needed to avoid the lost accuracy.

A false commit occurs when $y_i = 0$ but the learned router commits, and a false escalation occurs when $y_i = 1$ but the learned router escalates. Let $\rho = \Pr[y_i = 1]$ be the greedy-correct rate. We define $\mathrm{FPR}(\tau)$ as the conditional probability of a false commit among greedy-wrong problems, and $\mathrm{FNR}(\tau)$ as the conditional probability of a false escalation among greedy-correct problems. The probability that the learned router disagrees with the oracle is therefore

$$(1 - \rho)\,\mathrm{FPR}(\tau) + \rho\,\mathrm{FNR}(\tau).$$

The proposition uses an iso-accuracy accounting cost, not raw runtime cost. A false escalation wastes at most $K - 1$ samples relative to the oracle. A false commit spends fewer samples in raw compute, but it is charged by the extra SC budget that would have been needed to avoid the accuracy loss. Under this accounting, the per-problem regret is at most $(K - 1)$ whenever the learned router disagrees with the oracle, and is zero otherwise. Multiplying the disagreement probability by $K - 1$ and taking expectations gives the bound in Eq. (3). The $\mathcal{O}(N^{-1/2})$ term summarizes the standard finite-validation error incurred when selecting τ from empirical estimates. □

Role of AUC. The bound depends on the weighted false-commit and false-escalation rates at a chosen threshold. AUC provides a threshold-free diagnostic for the same ranking problem. After orienting the PDE+ score so that larger values indicate harder problems, AUC measures how often a greedy-wrong problem is ranked above a greedy-correct problem. Thus AUC is not used as a calibrated-probability claim. It is used to assess whether the prefill score provides an ordering from which a low-error routing threshold can be selected on validation data.

Relation to HATS. The proposition analyzes a simplified prefill-threshold router. HATS adds a greedy probe before committing: the scheduler returns the greedy trace only if the answer parses and was not truncated, and a borderline prefill score additionally requires the token-mean log-probability to pass the guard. These checks can turn a potential commit into an escalation, reducing false commits at the cost of more conservative routing. The simplified bound therefore captures the main dependency on prefill ranking quality, while the online scheduler adds safeguards needed to preserve the SC-4 accuracy target.

C Implementation details

PDE+ training details. PDE+ is a two-layer MLP with hidden dimensions 512 and 128, ReLU activations, and dropout 0.1. We train it with AdamW, learning rate 10^{-3}, batch size 64, and up to 30 epochs, selecting checkpoints by held-out AUC. The prefill embedding is the final-layer hidden state averaged over the last five prompt tokens; the deployed scorer uses no prompt-level text features.

PDE+ calibration protocols. On MATH-500, we report both the main calibrated operating point and a strict out-of-fold run, where each problem receives a PDE+ score from a model trained without that problem's fold. For GSM8K and the AIME 2022–2024 suite, the transfer rows in Table 4 apply the MATH-trained PDE+ with the deployed thresholds and use no target labels. Target-calibrated PDE+ estimators, refit on each target dataset with the same thresholds, appear only in the diagnostics of Figure 2 and Table 11. Replayed on the same pools, the target-calibrated estimator differs from the zero-shot rows by two problems on GSM8K and one on AIME, so per-dataset calibration is an option rather than a requirement.

Baseline implementation scope. For ESC, AC, and TVS, the stopping decision is a deterministic function of the ordered sample stream. We use trace-driven replay on the fixed SC-4 pool only as an offline diagnostic frontier. For selected thresholds that look promising in replay, we run fresh end-to-end probes to test whether they still save tokens when traces are generated online.

We adapt DSC and RPC to our online vLLM setup, rather than reproducing their original systems exactly. The released DSC system uses a batch difficulty pipeline, whereas our DSC-style row uses a lightweight prompt-feature prior (length, mathematical markup depth, and brace depth) to set Beta pseudo-counts for the dominant-answer posterior, with $\tau = 0.95$, $n_{\min} = 2$, and $n_{\max} = 16$. The released RPC system aggregates fixed-K pre-generated reasoning paths with a Weibull-mixture pruning model, whereas our RPC-style row uses vLLM `logprobs=1`, perplexity-weighted majority voting, and quantile-based path pruning at $q = 0.20$ with $n_{\max} = 16$. These rows test whether DSC-style priors or RPC-style pruning help in the same online generation setup as HATS.

Prefill-routing reproductions. ACTSC is reproduced with FFN activations at the last prompt token, its DSN gap selection applied within each fold, MATH difficulty-level labels, and its label-free mean-$P(\mathrm{hard})$ threshold. For the two choices its paper leaves open, we use the gap split levels in our released code and a greedy trace for the single-sample branch. LLM Router routes among K models. With two targets, one greedy trace or SC-4, it reduces to a prefill binary gate, which we reproduce with Fisher-score layer selection, PCA to 100 dimensions, and L2-regularized logistic regression on target correctness. The Adaptive Thinking-style executor maps the out-of-fold quartiles of the agreement probe to per-problem budgets from $1{,}024$ to $4{,}096$ tokens with one greedy trace per problem. All reproductions use the same cached traces, parser, and five-fold out-of-fold protocol as HATS.

Training PDE+ on the Adaptive Thinking target, SC agreement, instead of greedy correctness lowers the out-of-fold AUC from 0.727 to 0.670 and the token saving from 42.7% to 36.6% at the quantile-matched operating point. ReProbe is a step-level verifier for search-based scaling with no budget interface. Its closest adaptation keeps our MLP, protocol, and policy and changes only where the probe reads, comparing the prefill state with states taken after or pooled over the first 128 to 256 decoded tokens on a fresh greedy pool. Decoded-token probes rank better, with AUC up to 0.759 against 0.704 at prefill, yet all variants stay within two problems of each other in routed accuracy and within one point in saving. A mid-generation probe also has no score until 128 to 256 tokens have been decoded.

MSAP decision details. The greedy trace is always included in the majority-vote pool. HATS cannot commit early if the final answer cannot be parsed, if the trace was truncated at the generation cap, or, on the borderline channel ($\tau_{\mathrm{low}} \le d < 0.5$), if the token log-probability is unavailable or below τ_{lp}; in each case it continues sampling. The log-probability guard uses the mean log-probability over generated tokens in the greedy trace, excluding prompt tokens. Votes are counted after normalizing final-answer strings with the same parser used for evaluation. Ties are broken by generation order. If none of the generated traces contains a parsed answer, the implementation returns the first generated answer string, which is usually marked incorrect by the verifier.

Prompts and parsing. All methods use the same instruction prompt, asking the model to reason step by step and place the final answer in `\boxed{}`. Final answers are extracted with `math_verify` and checked with SymPy equivalence rules for common mathematical forms, including fractions, radicals, and π-expressions.

Table 8: Zero-shot transfer across models (accuracy gap relative to same-pool SC-4 / token saving). Each probe is trained on its own MATH-500 and applied to GSM8K and AIME 2022–2024 without target labels. The DeepSeek-R1-Distill-Qwen-7B MATH-500 entry is the replay row of Table 5.

Model	MATH-500 (in-domain)	GSM8K	AIME 2022–2024
DeepSeek-R1-Distill-Qwen-7B	−1.4 pp / 41.4%	−2.8 pp / 53.3%	−2.2 pp / 15.5%
DeepSeek-R1-Distill-Qwen-14B	−1.2 pp / 41.5%	−0.6 pp / 70.6%	−2.2 pp / 25.8%
DeepSeek-R1-Distill-Qwen-32B	−0.8 pp / 44.1%	−0.6 pp / 72.6%	−2.2 pp / 26.7%
DeepSeek-R1-Distill-Llama-8B	−1.0 pp / 34.4%	−2.8 pp / 65.4%	−2.2 pp / 17.7%
Qwen3-8B	−1.8 pp / 44.6%	−1.0 pp / 73.2%	+0.0 pp / 17.8%
Qwen3.5-9B	−0.6 pp / 49.4%	+0.0 pp / 76.3%	−3.3 pp / 35.6%
Qwen3.6-27B	−0.4 pp / 58.5%	−0.4 pp / 73.2%	−3.3 pp / 75.5%
Gemma-3-12B-IT	−1.0 pp / 51.7%	−0.8 pp / 73.2%	+0.0 pp / 31.7%

Table 9: Zero-shot transfer across datasets on DeepSeek-R1-Distill-Qwen-7B with the fixed MATH-500 configuration, sorted by allocated budget.

Dataset	n	Greedy acc.	Gap vs. SC-4	Token saving	Avg. samples
SVAMP	300	75.7%	−1.0 pp	57.7%	1.56
AMC23	40	90.0%	+0.0 pp	45.1%	1.70
GSM8K	500	73.0%	−2.8 pp	53.3%	1.80
MATH-500 (calibration source)	500	71.4%	−1.4 pp	41.4%	2.08
GSM-Hard	300	38.0%	−1.0 pp	29.7%	2.58
AIME 2025	30	36.7%	−3.3 pp	22.7%	2.70
AIME 2022–2024	90	42.2%	−2.2 pp	15.5%	3.00
MinervaMath	272	11.8%	+0.0 pp	12.6%	3.38

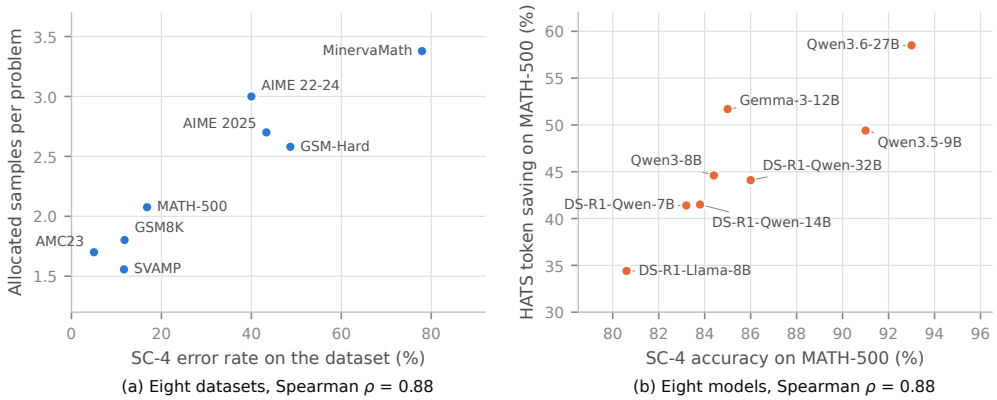
D Additional results and reproducibility

Zero-shot transfer. Tables 8 and 9 extend the protocol of Table 4. Each model uses a probe trained once on its own MATH-500 prefill states with greedy-correctness labels, the operating point follows the same source-percentile rule on that model's own scores, and the policy, guards, and vote rules are unchanged. The additional models and datasets are single generations under the SC-4 protocol, with the 24,576-token cap on AIME 2022–2024. Figure 3 shows the two regularities behind the tables. Across the eight datasets the allocated budget rises with difficulty (Spearman $\rho = 0.88$ against the SC-4 error rate). Across the eight models the MATH-500 saving rises with SC-4 accuracy ($\rho = 0.88$), and the strongest model, Qwen3.6-27B, allocates only 1.02 samples on AIME because its greedy pass already solves most problems. On the competition sets a gap of a few points corresponds to zero to three problems, and single runs on 90 problems are noisy. Regenerating the target-calibrated AIME configuration five times end to end, with both arms drawing from one trace stream per problem, gives a gap of -0.4 ± 0.5 pp. On MATH-500 itself HATS commits 58.2% of problems after one checked greedy trace, routes 19.4% through the small vote, and sends 22.4% to the full budget.

Low-budget baselines in the transfer settings. Because zero-shot HATS loses accuracy on GSM8K and AIME, Table 10 compares it with policies that need no prefill score, replayed on the same cached traces. These are fixed SC-k for $k \leq 4$, an agreement-first policy that decodes the greedy trace and one sample and fills the SC-4 budget only if they disagree, and the guard-only policy that commits whenever the validity guards pass, which is the constant-score ablation of Table 6. On GSM8K, HATS reaches SC-3 accuracy with 1.80 instead of 3 samples and twice the token saving, while the agreement-first policy keeps SC-4 accuracy at a smaller saving. On AIME, HATS sits between SC-3 and the agreement-first policy, which retains SC-4 accuracy at 8.9% saving. The prefill score buys accuracy over the guards alone and compute over fixed low budgets, and on hard sets a simple agreement rule remains competitive at the SC-4 accuracy level.

Table 11 reports a GSM8K length-abstention check with a target-calibrated estimator. Abstaining on long greedy traces recovers 0.6 points of accuracy at a small cost in saving and leaves most of the gap to SC-4 in place, so it is a mitigation rather than a fix.

Post-decode baselines. To test whether decoded agreement contains a useful stopping signal, we sweep post-decode rules over the cached SC-4 traces. The best AdaptiveConsistency point in this sweep, at threshold 0.60, reaches the SC-4 accuracy with 38.0% token saving. It is selected after all



(a) Eight datasets, Spearman $\rho = 0.88$ (b) Eight models, Spearman $\rho = 0.88$

Figure 3: The two regularities behind Tables 8 and 9. (a) On DeepSeek-R1-Distill-Qwen-7B the allocated budget rises with dataset difficulty. (b) Across eight models the MATH-500 token saving rises with the model's SC-4 accuracy, since stronger greedy passes make more problems commit-safe.

Table 10: Low-budget policies without a prefill score, replayed on the cached DeepSeek-R1-Distill-Qwen-7B traces (accuracy / avg. samples / token saving relative to SC-4). HATS entries are the zero-shot results of Table 4.

Policy	GSM8K ($n = 500$)	AIME 2022–2024 ($n = 90$)
SC-1 (greedy)	73.0% / 1.00 / 80.0%	42.2% / 1.00 / 71.7%
SC-2	81.2% / 2.00 / 53.4%	51.1% / 2.00 / 49.2%
SC-3	85.2% / 3.00 / 26.5%	54.4% / 3.00 / 24.1%
SC-4 (reference)	88.2% / 4.00 / 0.0%	60.0% / 4.00 / 0.0%
Agreement-first (2, then 4)	88.2% / 2.72 / 29.2%	60.0% / 3.40 / 8.9%
Guard-only commit, else SC-4	84.4% / 1.57 / 61.4%	55.6% / 2.37 / 29.5%
HATS, zero-shot	85.4% / 1.80 / 53.3%	57.8% / 3.00 / 15.5%

four traces exist, so it is an offline reference rather than an online baseline, and the same threshold run end to end does not reproduce the saving (Figure 4). The DynaThink row of Table 2 is a self-rating control, and its collapse to greedy decoding suggests that post-decode introspection cannot replace the prefill difficulty signal. Majority voting is not monotone in budget. Doubling SC-4 to SC-8 fixes 19 problems and breaks 18 (exact McNemar $p = 1.00$), and HATS-K8 differs from SC-8 by 18 fixed against 17 broken ($p = 1.00$). A paired bootstrap over the 500 shared MATH-500 problem IDs gives 95% confidence intervals of $[-4.8, -0.4]$ pp for HATS minus SC-4 accuracy, $[36.4, 44.0]\%$ for token saving, and $[45.0, 50.9]\%$ for sample saving.

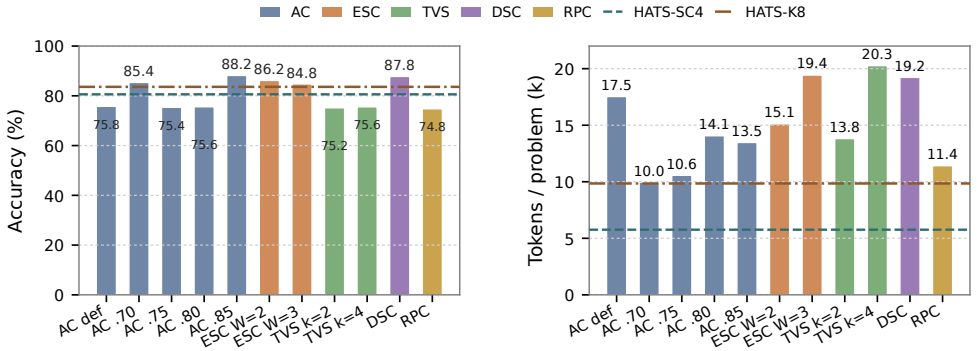


Figure 4: Completed end-to-end baseline operating points on MATH-500. Horizontal reference lines mark the HATS operating points calibrated to the SC-4 and SC-8 references.

Table 11: GSM8K length-abstention diagnostic ($n = 500$) with a target-calibrated PDE+ estimator and fixed MATH thresholds. Length abstention partially mitigates the accuracy loss but does not close the SC-4 gap.

Method / variant	Acc. (%)	Avg. samples	Avg. tokens	Token saving
SC-4 (fixed budget)	88.20	4.00	5,931	0.0%
HATS (fixed MATH thresholds)	85.80	–	2,788	53.0%
+ length abstain 0.60	86.40	1.91	3,010	49.3%
+ length abstain 0.50	86.00	1.95	3,099	47.8%

Calibration and estimator diagnostics. Table 3 reports calibration protocols for HATS, rather than separate methods. These checks test whether the MATH-500 savings depend on fitting the prefill estimator or selecting routing thresholds on the same benchmark.

In the strict out-of-fold setting, each MATH-500 problem receives a PDE+ score from a cross-validation model trained without that problem's fold. The downstream HATS thresholds are fixed before evaluation, rather than reselected on the held-out fold. Paired-bootstrap intervals for this run are $[-3.8, +0.6]$ pp for HATS minus SC-4 accuracy, 36.1%–43.8% for token saving, and 44.4%–50.4% for sample saving.

In the external-calibration setting, PDE+ is trained on a separate MATH-lighteval training subset, and τ_{low} is selected on a held-out calibration subset before evaluating MATH-500. This protocol uses no MATH-500 examples for PDE+ training or threshold selection. It is less token-efficient than benchmark calibration, but still preserves the SC-4 accuracy target.

Scoring and replay protocol. All runs, including the SC-4 and SC-8 references, are scored by the same checker with its SymPy LaTeX parser. Without that parser, answers that differ from the reference only in formatting, such as `\dfrac{4}{3}` against `\frac43`, count as wrong, which lowers the two references to 80.6% and 81.4%. Replay analyses (Tables 5, 12, 13, 14, and 10) re-execute the policy over the cached SC-4 traces, so commit decisions match the reported run and differences arise only from which cached traces fill the voting stage.

Threshold sensitivity and nested selection. In the reported run the probe scores were out-of-fold, but the thresholds were selected on the same 500 problems that are reported. Table 12 replays the policy over a 12×15 grid, $\tau_{\text{low}} \in [0.05, 0.60]$ and $\tau_{\text{lp}} \in [-0.80, -0.10]$ in steps of 0.05, and under a nested protocol that re-searches the thresholds for each evaluation fold on the remaining folds only. Accuracy stays within 81.8–82.0% in every cell and only the saving moves, because the guards block unsafe commits and a threshold change mainly shifts problems into the bounded majority-stop stage. The log-probability guard sits in a safe region, since no MATH-500 greedy trace has a mean log-probability below -0.40. Replaying the vote-stop parameters $(k_{\text{easy}}, k_{\text{hard}})$ over $\{2, 3\} \times \{2, 3, 4\}$ leaves accuracy unchanged in every variant and moves saving between 37.5% and 45.5%.

Table 12: Threshold sensitivity on the cached MATH-500 traces (replay, SC-4 reference 83.2%).

Setting replayed	Acc. (%)	Token saving
Deployed thresholds $(0.30, -0.40)$	81.8	41.4%
± 0.10 / ± 0.20 neighborhood (45 cells)	81.8–82.0	41.3–41.4%
All 180 grid cells	81.8–82.0	37.8–42.0%
Six vote-depth variants	81.8	37.5–45.5%
Fully nested per-fold re-search (3 seeds)	82.0	42.6–42.7%

Guard interceptions and training-set size. At the reported thresholds PDE+ scores 329 of 500 problems as easy and the guards intercept 60 of them (Table 13). Most intercepted greedy answers are wrong, and the escalated vote recovers about half of those. Table 14 retrains PDE+ on 10 to 100% of each fold's training split with three seeds and quantile-matched thresholds, so that only ranking quality varies, and replays the routing with the out-of-fold scores. The AUC elbow lies at roughly 100 to 200 labels per fold, and routing accuracy is already in band at 40 labels because the guards and staged voting absorb ranking noise.

Table 13: Guard interceptions on MATH-500 among the 329 problems scored easy.

Interception reason	Count	Greedy wrong
No parseable final answer	42	42
Truncated at generation cap	9	1
Fails SymPy parse guard	9	3
Mean log-prob below -0.40	0	0
Total	60	46 (76.7%)

Table 14: PDE+ training-set size (labels per fold, with AUC as mean $\pm$ SD over three seeds).

Labels	AUC	Acc. (%)	Saving
40	0.636 ± 0.032	81.3	44.9%
100	0.671 ± 0.016	81.4	43.9%
200	0.708 ± 0.011	81.8	39.9%
300	0.722 ± 0.014	81.9	42.5%
400	0.735 ± 0.004	81.7	42.7%

Estimator quality and realized saving. The routing-regret analysis predicts that prefill routing is useful when the score ranks greedy-wrong problems above greedy-correct ones, whether or not it is calibrated. Table 15 compares a non-informative score, the learned PDE+ score used by HATS, and an oracle score based on greedy correctness. The constant-score row shows that low-cost checked-greedy commits alone can save tokens but do not preserve the SC-4 target, whereas PDE+ recovers most of the Oracle-PDE saving while staying close to SC-4 accuracy. Table 16 adds prompt-level text features to the hidden-state input on a 100-problem subset. The estimator used in HATS reads hidden states only. The bucket diagnostics in Figure 2 provide the corresponding full-dataset view on MATH-500 and scoped transfer diagnostics on GSM8K/AIME.

Table 15: Estimator quality and realized compute saving on MATH-500 (AUC is strict out-of-fold).

Router	AUC	Acc.	Target met	1-sample	Samples	Token saving
Constant score (0.49)	0.50	79.00%	No	57.4%	1.90	46.1%
PDE+ & MSAP (HATS)	0.73	80.60%	Yes	58.2%	2.08	40.2%
Oracle-PDE (upper bound)	1.00	83.20%	Yes	67.2%	2.07	48.1%

Table 16: Feature-source ablation for PDE+ under MSAP routing on a 100-problem MATH-500 subset.

PDE+ features	n	Acc.	Samples	Tokens	Full SC-4 rate
Hidden states only	100	68.0%	2.25	6,051	21.0%
Text features only	100	69.0%	2.44	6,538	18.0%
Hidden states + text	100	70.0%	2.22	5,850	17.0%

Serving setup and single-stream profile. The serving runs of Table 7 use a stock vLLM OpenAI-compatible server with continuous batching and prefix caching. SC-4 submits the greedy and three stochastic traces of a problem concurrently; HATS keeps its greedy, guard, escalate order within a problem and draws its concurrency from overlap across problems. Throughput counts problems completed end to end per hour, so a problem may issue several generation calls. Figure 5 plots the concurrency sweeps on both GPUs. A repeat run on the RTX 3090 reproduces the gains, with $+16.0\%$ at concurrency 32 and $+58.7\%$ at 64. On the A800 the best SC-4 configuration is at concurrency 128 while HATS keeps scaling to 384. At their best configurations HATS completes 38.2% more problems per hour at 26.3% lower GPU time per correct answer, and the accuracy difference is not significant ($p = 0.12$).

At inference time, PDE+ reuses the reasoner's prefill hidden states with only a small MLP scorer. Table 17 profiles the single-stream prototype, where HATS emits fewer tokens than SC-4 but is slower because it interleaves a greedy probe, guard checks, and small adaptive voting calls with the GPU idle in between. In a small eight-problem batching check, HATS is faster than SC-4 when each problem is served alone (55.6 against 75.4 s per problem) but slower when the eight problems are batched (27.7 against 17.7 s), because fixed-budget sampling gains more from batching. Under continuous batching the idle time between HATS calls is filled by other problems, which is why the gains in Table 7 exceed what this profile suggests.

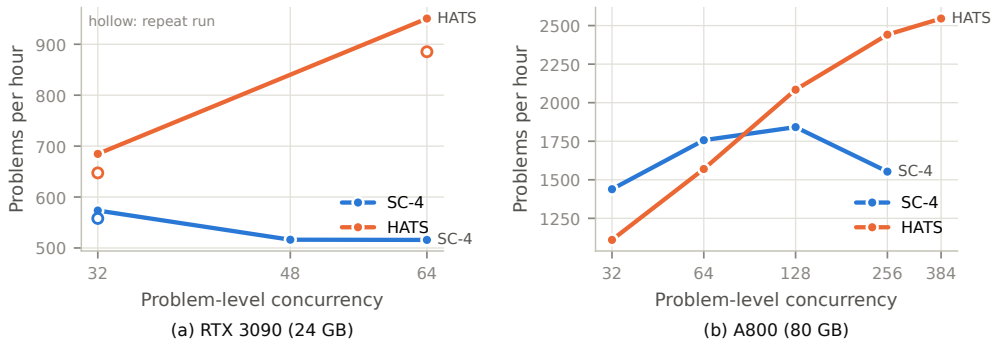


Figure 5: Throughput against problem-level concurrency under continuous batching. SC-4 peaks at concurrency 32 on the RTX 3090 and 128 on the A800, whereas HATS, holding fewer KV-cache slots per problem, keeps scaling; hollow markers are the repeat run.

Table 17: Hardware profile on a 100-problem MATH-500 subset.

Method	sec/prob	tokens	tok/s	mem
Greedy	48.72	2,248	46.2	22,236
SC-4	102.20	8,837	86.3	22,276
HATS	113.62	5,577	49.4	22,272

E Reproducibility

A supplementary code package accompanies the paper. It contains the HATS/MSAP runner, PDE+ feature extraction and training scripts, fixed-budget SC baselines, online post-decode baseline runners, dataset loaders, requirements, and small JSON artifacts for checking the main MATH-500 runs.

Dependencies. The experiments use Python 3.11, PyTorch, vLLM, Hugging Face Transformers and Datasets, NumPy, scikit-learn, SymPy, and `antlr4-python3-runtime`. Install via:

```
pip install -r requirements.txt
```

Model checkpoints and large third-party datasets are loaded from their public releases and are not redistributed.

Running HATS. The main entry point is `hats.py`. Example command for the MATH-500 HATS run:

```
python hats.py --model deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
  --dataset math500 \
  --pde_predictions data/pde_math500_predictions.json \
  --out results/hats_math500.json
```

The package README also lists commands for SC-K, PDE+ training, strict OOF checks, and online post-decode baselines.

Computational cost. The main MATH-500 runs use one 24 GB GPU with vLLM. Included JSON artifacts allow the main HATS output to be checked without rerunning all prefill feature extraction and generation steps.

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 clearly state the problem setting, the proposed method, the main empirical findings, and the scope of the contribution. The claims are supported by the results and analyses reported in the paper.

   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: Please refer to Section 7.

   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: Please refer to Section 5, Appendices A and B.

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: The paper reports the models, datasets, evaluation protocol, baselines, calibration variants, and uncertainty estimates needed in Section 6; additional implementation and audit details are provided in the Appendices.

   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: The supplementary material includes the code, experiment data, and instructions needed to reproduce the main results. The benchmark datasets are public and are described in Section 6.

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 specifies the datasets, models, decoding settings, metrics, and baseline protocols. Appendices C and D give PDE+ training details, calibration procedures, MSAP decision rules, and hardware settings.

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: We report paired-bootstrap confidence intervals for the main comparison in Appendix D, with the corresponding result summarized in Section 6. Serving comparisons report McNemar tests on accuracy, training-size results report mean and standard deviation over three seeds, and gaps on small competition sets are stated as problem counts.

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: We report the single-24GB-GPU vLLM setup for the main experiments in Section 6, the RTX 3090 and A800-80GB serving configurations in Section 6 and Appendix D, and memory and per-problem timing for the single-stream profile in Appendix D.

   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 work uses public math benchmarks and open-weight models, and does not involve human subjects, private data, or deployment decisions affecting individuals.

   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: Please refer to the Broader impact paragraph in Section 7.

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: This paper poses no such risks.

    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: We credit the public datasets, models, baselines, and libraries used in the experiments, and use them according to their public terms and licenses.

    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: This paper does not release new assets.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: [Yes]

    Justification: The paper studies test-time trace allocation for LLM reasoners. Publicly available LLM checkpoints are used as the base reasoners, and HATS uses their prefill hidden states, generated traces, and token log-probabilities for routing decisions. These uses are described in the method and experimental setup sections.

    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.