RACE: CALIBRATED KV-CACHE RESERVATIONS FROM PREFILL STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM serving systems make request-admission decisions before the decoding demands of incoming requests are fully known. In prefill–decode disaggregated systems, forecasting output length can help decoding workers plan key–value cache reservations and determine how many requests to admit. The challenge is that repeated generations from the same prompt can differ in length, making reservations based on a single predicted length vulnerable to unexpectedly long responses. We introduce Rollout-Aware Calibrated Endpoints (RACE), which predicts a token budget by accounting for both typical response length and response-length variability. RACE learns these quantities with two predictors trained on prompt-processing hidden states and multiple sampled responses per training prompt. It assigns larger safety margins to prompts with more variable response lengths, then calibrates these margins on an independent prompt set. On Qwen3-8B with DeepScaleR, RACE reduces the average token budget by 9.94% compared with GBM-Quantile-PAC, a calibrated quantile-regression baseline using the same features. Its budgets accommodate 97.56% of test outputs under the configured output-length limit. These smaller reservations reduce the capacity set aside per request, leaving more room for concurrent work under reservation-based scheduling.

1 INTRODUCTION

LLM serving systems share key–value (KV) cache capacity among concurrent requests, and each request uses more of this capacity as its response grows. In prefill–decode disaggregated systems, forecasting response length can help decoding workers decide which requests to admit. A reservation-based scheduler assigns each request an output-token budget: the number of generated tokens for which it plans cache space (Kwon et al., 2023; Zhong et al., 2024; Qin et al., 2025). Reserving space for the maximum permitted output length leaves capacity unused when responses finish much earlier, reducing the space available for other requests. Smaller, request-specific reservations can reduce this waste, but an underestimated response will need cache space beyond its initial reservation (Jin et al., 2023). The budget guides capacity planning; the service's output-length limit determines when generation must stop. Our goal is to reduce the space reserved per request while keeping the proportion of responses that need more space than reserved below a specified level.

Previous work uses response-length predictions to make concrete scheduling decisions. S3 (Jin et al., 2023) estimates output lengths to plan cache use and select requests to run together. Learning-to-rank schedulers (Fu et al., 2024) identify requests likely to produce shorter responses and prioritize them to reduce waiting time. ProD (Wang et al., 2026a) improves length estimation by training on multiple responses to each prompt, providing a more stable estimate of its typical response length. For cache reservation, however, estimating the typical length leaves another decision: how much extra space should be added for responses that run longer? H-MAS (Liu et al., 2026) estimates this extra space from past prediction errors, grouping requests by input-prompt length and adding the same margin within each group. This shared margin cannot distinguish prompts whose response lengths are consistent from those whose lengths vary widely. Under stochastic decoding, multiple responses to a training prompt reveal these differences. We investigate how to use that information to learn a separate safety margin for each request, allocating more space where response lengths vary more and avoiding unnecessary reservations where they are more consistent.

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

We introduce **R**ollout-**A**ware **C**alibrated **E**ndpoints (RACE), a method that predicts reservation budgets from prefill states for LLM services using stochastic decoding. Even with fixed temperature and other sampling settings, independently sampled responses to the same prompt can follow different generation paths and terminate at different lengths. The amount of variation also differs across prompts. Two prompts with similar typical response lengths may therefore need different safety margins. RACE uses repeated responses to learn this distinction, allocating larger margins to prompts with more variable lengths and smaller margins to those with more consistent lengths.

RACE separates budget construction into learning the allocation of safety margins and calibrating their overall size. Using hidden states computed during prefill, two predictors estimate a prompt's typical response length and its response-length variability. Their training targets come from the median and spread of multiple sampled responses to each training prompt. RACE adds a margin proportional to the predicted variability to the predicted typical length. An independent set of calibration prompts then determines a shared multiplier for these margins to meet a specified population-coverage target. This division lets repeated-response supervision shape how capacity is distributed across requests, while calibration sets the overall level of protection against budget overruns. At inference time, RACE computes the budget from the incoming prompt's prefill states, making it available before decoding begins.

On Qwen3-8B (Yang et al., 2025) with DeepScaleR (Luo et al., 2025), RACE reduces the average number of output tokens reserved per request by 9.94% relative to GBM-Quantile-PAC, a calibrated quantile-regression baseline using the same features. Its reservations accommodate 97.56% of test responses under the service's output-length limit, 0.59 percentage points higher than the baseline. RACE therefore reserves less space per request while accommodating a slightly larger fraction of responses. An ablation further demonstrates the benefit of tailoring safety margins to individual requests: compared with a variant that randomly reassigns predicted length variability across prompts, RACE reduces the average output-token reservation by 13.19% and increases observed coverage by 0.95 percentage points.

In scheduling simulations, RACE's smaller reservations allow waiting requests to start earlier. Across the evaluated cache capacities and request arrival rates, RACE reduces mean request completion time by 21.44% and increases simulated token throughput by 4.01% relative to GBM-Quantile-PAC. These results connect the reduction in reserved capacity to earlier admission and faster completion in the simulated system.

Our main contributions are as follows:

- We propose RACE, a method for predicting request-specific KV-cache reservations from prefill states. RACE learns typical response length and length variability from repeated training responses, uses these predictions to assign safety margins, and independently calibrates the resulting reservations to a specified coverage target.
- We evaluate RACE across three LLMs and three datasets. Across the four evaluated settings, RACE reserves between 9.94% and 22.26% fewer output tokens than GBM-Quantile-PAC, while its observed response coverage exceeds 95% in every setting.
- We demonstrate the benefit of learned safety margins with the length predictor and training cohort held fixed. With three responses per training prompt, RACE reduces reservations by 13.19% and increases observed coverage by 0.95 percentage points relative to shuffled margins. This advantage also holds with two, five, and eight responses per prompt.

2 Problem Formulation

Output length and the predicted budget. RACE predicts how many output tokens a serving system should reserve KV-cache capacity for before decoding begins. We consider a fixed LLM M and a fixed stochastic decoding policy π. For a prompt X, prefill produces hidden states $S = h_M(X)$. Let $L = L(X, M, \pi, \Omega)$ denote the number of output tokens generated before natural termination, where Ω represents the randomness of token sampling. With a service output-length limit of H tokens, the response length is

$$Y^H = \min(L, H). \tag{1}$$

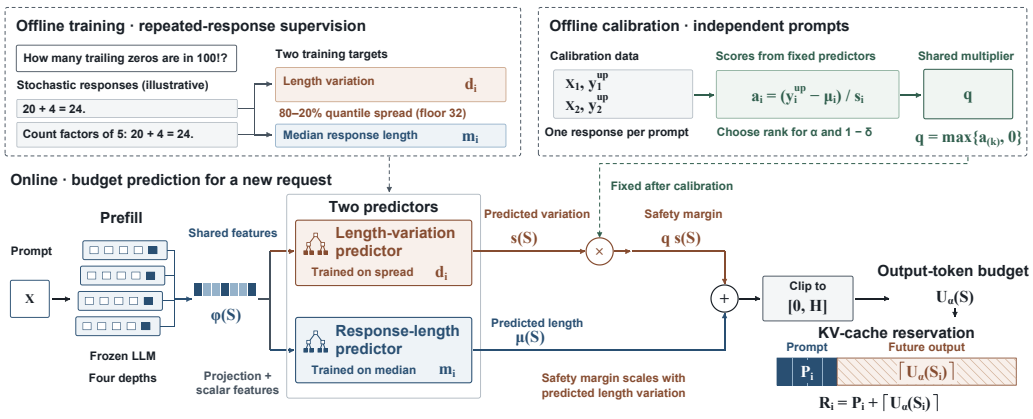

Figure 1: **Overview of RACE.** Two predictors use prefill features to estimate response length and its variation across repeated generations. Multiple responses per training prompt provide the training targets. RACE assigns larger safety margins to requests with greater predicted length variation and calibrates their size on a separate set of prompts to meet the chosen coverage target. Adding the calibrated margin to the predicted response length gives the output-token budget for KV-cache reservation.

RACE uses S to predict an output budget $U_\alpha(S) \in [0, H]$. This budget specifies the capacity planned for the response, while H specifies the maximum number of output tokens the service will generate.

Coverage and reservation size. A predicted budget covers a response when the response length does not exceed that budget. For a chosen coverage target α, we require

$$\Pr\{Y^H \leq U_\alpha(S)\} \geq \alpha, \tag{2}$$

where the probability is over incoming prompts and the responses sampled for them under policy π. For example, $\alpha = 0.95$ sets a target of accommodating at least 95% of responses across this request population. Our objective is to reduce the average output-token budget $\mathbb{E}[U_\alpha(S)]$ while meeting the coverage target. These two quantities capture the trade-off introduced above: smaller budgets reserve less capacity, whereas higher coverage means fewer responses need space beyond their initial reservations.

Using the budget for KV-cache reservation. For a request with P_i prompt tokens, the planned KV-cache reservation, expressed in token slots, is

$$R_i = P_i + \lceil U_\alpha(S_i) \rceil. \tag{3}$$

The first term accounts for the prompt's KV cache; the second reserves space for the predicted output budget. A reservation-based scheduler uses R_i to decide whether the request fits within the available capacity. At the same coverage level, smaller average budgets reduce the capacity set aside per request and can allow more requests to be admitted. The next section describes how RACE learns safety margins from repeated responses and calibrates their size to meet the coverage target.

3 RACE

RACE predicts an output-token budget using two predictors with prefill states S as input. One estimates response length $\mu(S)$; the other estimates how much response lengths vary across repeated generations of the same prompt, $s(S)$. RACE adds a safety margin to $\mu(S)$, with larger margins for greater predicted length variation. Calibration on independent prompts and responses adjusts these margins to meet the chosen coverage target.

3.1 EXTRACTING PREDICTION FEATURES DURING PREFILL

RACE reuses hidden states computed by the target model while processing the prompt, making the prediction features available before decoding begins. For a model with D layers, RACE reads the last prompt-token state at depths nearest $D/4$, $D/2$, $3D/4$, and D. A shared, fixed Gaussian projection maps each state to a lower-dimensional vector, forming the projected features used to predict response length and its variability.

Alongside the projected states, RACE provides the predictors with explicit measurements of the input and hidden representations. These include the prompt's token count and the magnitude of each selected state, measured by its L2 norm. RACE also computes cosine similarity and Euclidean distance between states at successive selected depths. Cosine similarity measures directional agreement between the states, while Euclidean distance measures the size of their difference.

RACE further computes token-specific scores for response termination, reasoning boundaries, and line breaks. At each selected depth, it applies the model's output-layer weights to the normalized state and retains the scores for EOS, `<think>`, `</think>`, and newline. We concatenate these scalar measurements with the projected states to form the feature vector $\phi(S)$, which serves as the input to both predictors. Appendix B lists the feature definitions, normalization conventions, and input dimensions.

3.2 LEARNING RESPONSE LENGTH AND LENGTH VARIATION

Prompts with similar median response lengths can differ in how much their sampled response lengths vary. RACE learns this distinction from multiple responses sampled for each training prompt under the same decoding policy.

For training prompt X_i, let $S_i = h_M(X_i)$ denote its prefill states. Let O_i be the list of observed lengths of its naturally terminated sampled responses, retaining repeated length values. Responses stopped at the collection limit are excluded from these training targets because their naturally terminated lengths are unknown. Each prompt with $|O_i| \geq 2$ contributes one pair of training targets:

$$m_i = \text{median}(O_i), \qquad d_i = \max\{Q_{0.8}(O_i) - Q_{0.2}(O_i), 32\}. \tag{4}$$

Here Q_p denotes the empirical p-quantile. The median m_i is the response-length target. The difference between the 80th and 20th percentiles measures how much the observed response lengths vary, with a 32-token floor keeping the variation target positive. Each eligible prompt contributes one training example, regardless of the number of naturally terminated responses available for that prompt.

The two LightGBM regressors take the same feature vector $\phi(S_i)$ extracted from S_i as input. They minimize L1 loss on $\log(1 + m_i)$ and $\log(1 + d_i)$, respectively. For a new request with prefill states S, inverse transformation and clipping produce a nonnegative response-length estimate $\mu(S)$ and a positive length-variation estimate $s(S)$, both in token units. Appendix B specifies these transformations and clipping ranges. Model selection uses held-out validation prompts.

3.3 CALIBRATING THE SAFETY MARGIN

With the two predictors fixed, RACE adds a safety margin $q\, s(S)$ to the length estimate $\mu(S)$. The predicted variation $s(S)$ determines how the margin differs across requests, while calibration selects the shared coefficient $q \geq 0$. We use one prespecified sampled response from each of n independent calibration prompts, held out from fitting, validation, and testing.

For calibration prompt i, let $\mu_i = \mu(S_i)$ and $s_i = s(S_i)$. We set the target y_i^{up} to the observed response length when generation ends naturally, and to H when generation is stopped by the data-collection limit. This provides an observed value or a conservative upper bound for the service response length Y_i^H. We then compute

$$a_i = \frac{y_i^{\text{up}} - \mu_i}{s_i}. \tag{5}$$

A positive score specifies how many multiples of s_i must be added to μ_i to cover the target. A nonpositive score means that the length prediction already covers it.

To select a multiplier for future requests, we sort the scores as $a_{(1)} \leq \cdots \leq a_{(n)}$. The coverage target α specifies the fraction of responses that the budgets should accommodate. The confidence level $1 - \delta$ accounts for uncertainty from using a finite calibration sample. We choose the smallest rank $k \in \{1, \ldots, n\}$ satisfying

$$\Pr\{Z \geq \alpha\} \geq 1 - \delta, \qquad Z \sim \mathrm{Beta}(k, n + 1 - k). \tag{6}$$

Because the ordered scores increase with k, choosing the smallest admissible rank keeps the budgets as small as this calibration criterion permits for the fixed predictors.

If no rank satisfies Equation 6, RACE returns the full service budget H. Otherwise, the selected score gives $q = \max\{a_{(k)}, 0\}$, and RACE predicts

$$U_\alpha(S) = \mathrm{clip}\big(\mu(S) + q\, s(S), 0, H\big). \tag{7}$$

Taking the maximum with zero keeps the added margin nonnegative. Clipping limits the reservation to the maximum output length permitted by the service.

When calibration and future requests are independent draws from the same population under the same decoding policy, the resulting budget rule satisfies Equation 2 with probability at least $1 - \delta$ over the calibration sample. Appendix C provides the proof.

4 Evaluating the Scheduling Benefits of RACE

We use a scheduling simulation to quantitatively evaluate whether RACE's smaller reservations allow waiting requests to start earlier and improve overall decoding throughput. The simulation models requests sharing limited KV-cache capacity, with predicted token budgets determining when each request can begin decoding.

Scheduling with predicted budgets. The scheduler uses the reservation R_i defined in Equation 3, which includes space for the prompt and the predicted output-token budget. Let K denote the total capacity in token slots, $\mathcal{A}$ the set of requests currently decoding, and $S_{\max}$ the maximum number of concurrent requests. The scheduler scans the waiting queue in arrival order and starts the first request j satisfying

$$\sum_{i \in \mathcal{A}} R_i + R_j \leq K, \qquad |\mathcal{A}| < S_{\max}. \tag{8}$$

The first condition checks whether the request's reservation fits within the remaining capacity; the second checks whether a decoding slot is available. Each reservation remains fixed until the request finishes, when its reserved capacity is released. A smaller reservation can therefore make room for a waiting request to start while other requests are still decoding.

Measuring scheduling outcomes. Each active request generates one output token per simulation step until it reaches its recorded response length. We measure token throughput as the total number of output tokens divided by the simulation duration. Job completion time is the number of steps from a request's arrival to its completion, including time spent waiting in the queue.

We also track the KV-cache demand implied by the generated tokens. After request i has generated $G_i(t)$ output tokens, it occupies $P_i + G_i(t)$ token slots in the simulation. Requests continue decoding if their outputs exceed the predicted budgets. We record the fraction of simulation steps in which the combined demand of active requests exceeds K, referred to as the capacity-excursion fraction.

Simulation protocol. We replay a fixed trace of 512 requests, with one recorded output length per request and an associated output-token budget from each method. We compare RACE and the baselines across multiple cache capacities and request arrival rates. Within each comparison, all methods receive exactly the same request order, arrival times, and output lengths, and use the same scheduling rule; only their per-request reservation budgets differ. Appendix E specifies the parameter values, arrival generation, and replay implementation in detail.

5 EXPERIMENTAL SETUP

Models and datasets. Our primary experiment uses Qwen3-8B on DeepScaleR (Luo et al., 2025), with 1,536 development prompts and 512 test prompts selected after deduplication. The development set contains 1,074 fitting, 185 validation, and 277 calibration prompts. We sample three responses per development prompt and two per test prompt. Additional feature-matched comparisons and repeated-response experiments use 500 new, nonoverlapping DeepScaleR prompts with four test responses each. We also evaluate RACE on DeepSeek-V2-Lite (DeepSeek-AI et al., 2024)/DeepScaleR, Qwen3-30B-A3B/GSM8K (Cobbe et al., 2021), and Qwen3-8B/ShareGPT (Kong et al., 2026), using predictors and calibration fitted for each setting. Table 11 summarizes these settings; Appendix B.3 describes data preparation and splitting.

Baselines. We compare RACE with four calibrated output-token budget predictors. GBM-Quantile-PAC (GBM-PAC) fits seven separate quantile regressors using the same prefill features as RACE. Global-PAC assigns every request the same token budget, obtained from the calibration response lengths. Mixture-PAC, our distributional baseline, fits a neural log-logistic mixture to repeated response lengths. Its primary configuration uses prefill features and additional vocabulary-level statistics; we also evaluate an alternative version using exactly the same features as RACE. AFT-PAC fits an accelerated-failure-time model to response lengths, including responses stopped at the collection limit; Appendix D.1 gives its fitting and calibration details. Reserving the full service limit H for every request provides a simple fixed-budget reference.

Implementation and parameters. For the primary Qwen3-8B/DeepScaleR experiment, generation uses temperature 0.6, top-p 0.95, top-k 20, and a collection and service limit of $H = 32{,}768$ output tokens. RACE projects each selected prefill state to 256 dimensions and combines the projected states with scalar features, giving 1,059 inputs to each of its two LightGBM regressors. The primary comparisons use a 95% coverage target and 95% calibration confidence. Calibrated methods share the same calibration prompts, prespecified responses, and treatment of responses stopped at the collection limit. Appendix B gives the feature definitions and predictor hyperparameters.

Evaluation metrics. Our primary metrics are service coverage and the mean number of output tokens reserved per request. Service coverage is the fraction of responses whose lengths do not exceed their predicted budgets. We also report excess and deficit reservation tokens for naturally terminated responses. For the primary RACE–GBM-PAC comparison, comparable coverage requires the lower bound of the 95% confidence interval for the coverage difference to be at least -1 percentage point. These intervals use 20,000 paired prompt-level bootstrap samples, keeping responses to the same prompt together. Scheduling simulations report token throughput, request completion time, and the fraction of simulation steps exceeding the KV capacity.

6 RESULTS

6.1 MAIN RESULTS

Table 1 summarizes the main results on three LLMs and three datasets, covering mathematical reasoning and conversational requests. Across all four settings, RACE reduces the average number of output tokens reserved per request by between 9.94% and 22.26% relative to GBM-PAC. Its predicted budgets accommodate more than 95% of test responses in every setting. RACE therefore achieves smaller reservations across the evaluated models and datasets while meeting the coverage target on the test responses.

On Qwen3-8B with DeepScaleR, RACE reduces the average reservation from 24,118 to 21,721 output tokens compared with GBM-PAC, while increasing coverage by 0.59 percentage points to 97.56%. The reservation savings exceed 20% on both DeepSeek-V2-Lite with DeepScaleR and Qwen3-8B with ShareGPT, reaching 22.26% and 20.55%, respectively. These smaller reservations leave more capacity available for additional requests under reservation-based scheduling. Table 4 provides the paired coverage comparisons and confidence intervals.

Table 1: **Reservation quality across four model–task settings.** Budget is mean reserved output tokens; Cov. is coverage (%). Calibrated methods use 95% coverage and 95% confidence. Blue/orange indicates smaller/larger budgets than GBM-PAC, with shading saturating at a 25% change. Bold marks the smallest budget in each setting.

	Qwen3-8B DeepScaleR		DeepSeek-V2-Lite DeepScaleR		Qwen3-30B-A3B GSM8K		Qwen3-8B ShareGPT	
Method	Budget↓	Cov.	Budget↓	Cov.	Budget↓	Cov.	Budget↓	Cov.
Service limit	32,768	100.00	65,536	100.00	32,768	100.00	32,768	100.00
Global-PAC	28,367	96.48	10,382	98.05	3,456	95.87	7,055	97.75
GBM-PAC	24,118	96.97	10,559	98.24	3,377	97.19	5,296	97.46
AFT-PAC	22,590	97.46	11,834	99.02	3,101	96.55	5,036	97.07
Mixture (matched)	22,380	97.36	9,006	98.24	3,225	96.59	4,682	96.39
Mixture (native)	**21,552**	97.85	8,615	98.44	3,089	96.44	4,505	96.48
RACE	21,721	97.56	**8,209**	98.44	**2,976**	96.47	**4,208**	96.48
Saving vs. GBM	9.94%		22.26%		11.87%		20.55%	

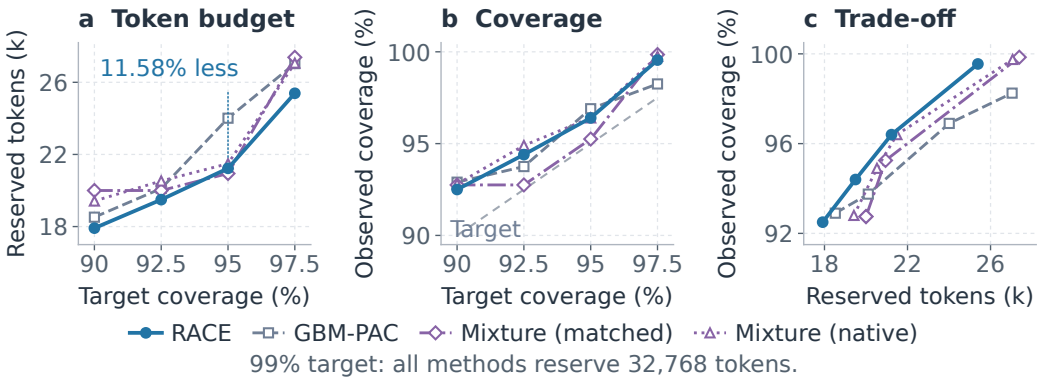

Figure 2: **Token reservations and observed coverage at different coverage targets.** The predictors remain fixed, and each method is recalibrated at 95% calibration confidence. All settings are evaluated on the same additional 500 prompts and 2,000 sampled responses. (a) Average output tokens reserved per request. (b) Percentage of responses that fit within their predicted budgets; the diagonal indicates equality between observed and target coverage. (c) Observed coverage plotted against average reservation. Lines connect the four evaluated coverage targets. At the 95% target, RACE reserves 11.58% fewer tokens than GBM-PAC.

6.2 Token Budgets at Different Coverage Targets

The main experiment uses a 95% coverage target. We next test whether RACE's reservation savings persist when this requirement changes. We evaluate coverage targets of 90%, 92.5%, 95%, and 97.5%, recalibrating each method without retraining its predictors.

Figure 2 shows how the predicted budgets change with the coverage target. Raising the target from 90% to 97.5% increases RACE's average reservation from 17,913 to 25,391 output tokens, while its observed coverage rises from 92.50% to 99.55%. Across all four targets, RACE reserves fewer tokens than GBM-PAC, and its observed coverage exceeds the corresponding target. At the same 95% target, RACE reduces the average reservation by 11.58%, with observed coverage of 96.40% for RACE and 96.90% for GBM-PAC. These results show that RACE meets different coverage requirements with smaller token reservations than GBM-PAC, leaving more capacity available for other requests. Appendix D.9 provides the complete results.

6.3 Can RACE Predict Response-Length Variation?

RACE uses its prediction of length variation, $s(S)$, to assign different safety margins to different requests. We test whether larger values of $s(S)$ correspond to larger differences between the lengths of responses generated from the same prompt. We rank the 512 primary test prompts by $s(S)$ and divide them into four equal groups. For the 496 prompts whose two sampled responses both terminate naturally, we compare the lengths and measure the absolute error of the length predictor $\mu(S)$.

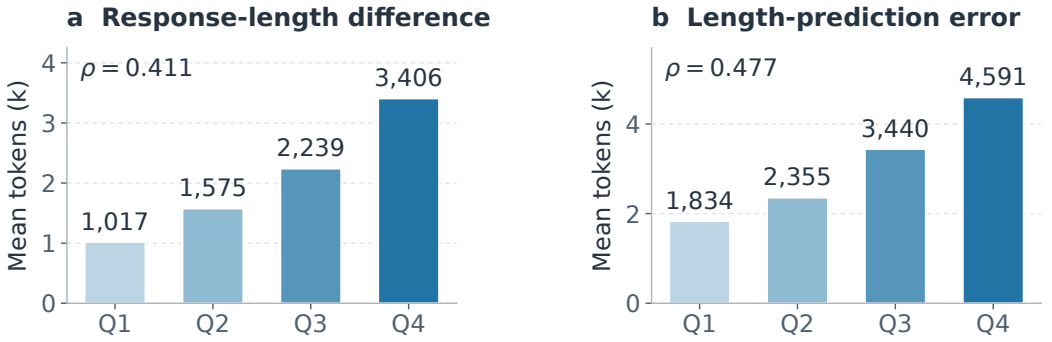

Figure 3: **RACE identifies requests with greater response-length variation.** This post-test analysis groups the 512 primary test prompts by predicted $s(S)$, from lowest (Q1) to highest (Q4). Panel (a) shows the mean absolute difference between two sampled response lengths. Panel (b) shows the mean absolute error of the length predictor $\mu(S)$, averaged over both responses. Both panels use naturally terminated pairs: 128, 127, 123, and 118 prompts in Q1–Q4. ρ denotes Spearman correlation across all 496 such pairs.

Figure 3a shows that the mean difference between the two response lengths increases from 1,017 tokens in the lowest predicted variation group to 3,406 tokens in the highest, a factor of 3.35. Across individual prompts, predicted variation is positively associated with the observed length difference (Spearman $\rho = 0.411$). The mean absolute error of $\mu(S)$ also increases across these groups, from 1,834 to 4,591 tokens (panel b). Together, these results show that RACE can identify, before decoding, prompts whose responses vary more in length and deviate further from a single length estimate. This ability provides a basis for assigning larger safety margins to those prompts instead of reserving the same extra space for every request. The next experiment evaluates the reservation savings from this assignment. Table 12 reports the group-level statistics.

6.4 Do Learned Safety Margins Reduce Token Reservations?

The previous experiment shows that RACE can identify prompts with greater response-length variation. We now test whether using these predictions to set safety margins reduces token reservations. We keep the length predictor fixed and compare RACE with three alternatives: adding the same margin to every request, making the margin proportional to predicted response length, and randomly reassigning RACE's predicted variation values to other requests. Each alternative is calibrated separately using the same calibration prompts and coverage target.

With three sampled responses per training prompt, RACE reserves 21,542 output tokens per request on average and covers 96.90% of test responses. Randomly reassigning the variation predictions increases the average reservation to 24,816 tokens and lowers coverage to 95.95% (Figure 4). RACE therefore reserves 13.19% fewer tokens while covering 0.95 percentage points more responses. This comparison shows that matching the predicted variation to the corresponding request improves both reservation size and coverage.

We repeat the comparison by training the variation predictor with two, three, five, and eight responses per prompt. RACE consistently outperforms random reassignment, reducing reservations by between 8.97% and 13.19% and increasing coverage by between 0.55 and 1.30 percentage points. These results show that RACE can learn effective safety margins from only a few responses per prompt, saving token capacity by adapting the margins to individual requests.

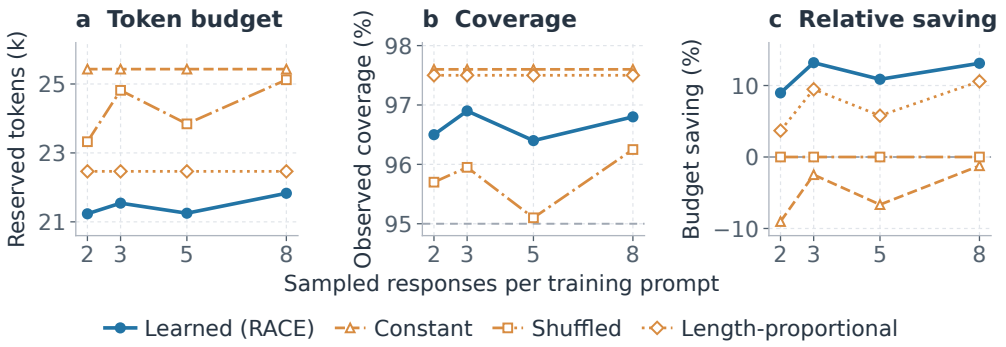

Figure 4: **Assigning the safety margin to the right prompt.** The length predictor and eligible training cohort are fixed. Each margin rule is separately calibrated, then evaluated on the same 2,000 outputs. Panels compare mean reservation, coverage, and budget savings relative to shuffled margins at each training-response count.

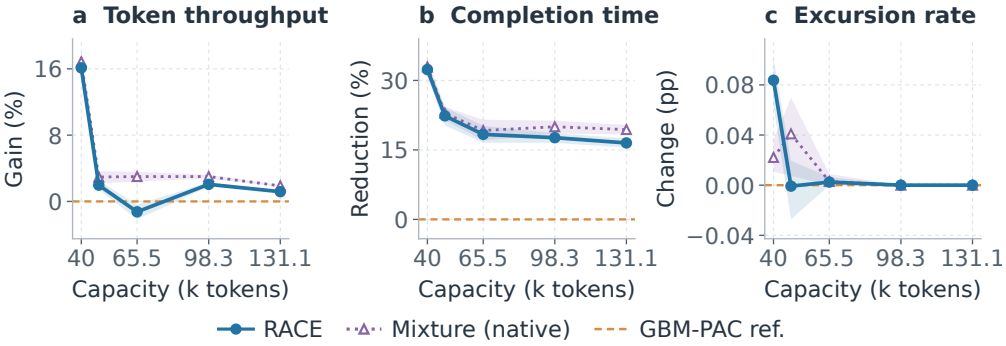

Figure 5: **Scheduling gains depend on available KV capacity.** Policies share the same 512 requests and arrivals. Curves show changes relative to GBM-PAC, averaged over five offered loads at each capacity. Bands are pointwise 95% arrival-seed-block bootstrap intervals. Positive values indicate higher throughput (a), shorter completion time (b), and more over-capacity steps (c).

6.5 Higher Throughput and Shorter Request Completion Time

Smaller reservations allow waiting requests to start sooner. Across the 25 capacity–load settings, RACE improves token throughput by 4.01% and reduces mean completion time by 21.44% relative to GBM-PAC (Table 17). Figure 5 shows where those aggregate gains arise. At 40,000 tokens of capacity, throughput increases by 16.12% and mean completion time falls by 32.35%: when space is scarce, saving tokens on each admission has greater scheduling value. The fraction of steps exceeding nominal capacity increases by 0.084 points at 40,000 tokens and by 0.017 points across the full grid. The joint view ties the reservation decision to its operational consequences: waiting time and throughput improve, while capacity overruns remain a separate quantity to monitor. Appendix E gives the paired complete results.

7 Conclusion

RACE learns response length and its variation from prefill states, then calibrates request-specific token budgets on independent prompts. Across the four evaluated settings, RACE reduces the average reservation relative to GBM-PAC while achieving observed coverage above 95%. With the length predictor fixed, learned safety margins improve both reservation size and observed coverage relative to shuffled margins across all four evaluated training-response counts. Scheduling simulations further show how smaller reservations help waiting requests start earlier and reduce completion time under limited KV-cache capacity. Together, these results demonstrate the value of learning how safety margins differ across requests and calibrating their overall size.

AI USE STATEMENT

In this work, we used AI-tools to assist with the implementation of our algorithms through iterative code revision and refinement. Additionally, we used AI-tools to polish the language and improve the clarity and readability of the manuscript. We have reviewed all AI-assisted work. Specifically, the authors reviewed the textual revisions and manually inspected and tested the modified code for correctness. We take responsibility for the final content of this work, including the text, claims, code, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies response-length prediction and calibrated KV-cache reservations to improve resource allocation in LLM serving. Although LLMs are trained on large amounts of Internet text that may contain harmful content, our study targets decoding length prediction understanding rather than direct deployment, which greatly reduces the risk of propagating biased information. All models and datasets used in our experiments are open-sourced and publicly available, ensuring transparency and minimizing potential ethical concerns.

REPRODUCIBILITY STATEMENT

We describe the RACE method in Section 3 and provide the feature definitions, predictor training settings, and data preparation procedures in Appendix B. Section 5 specifies the models, datasets, baselines, sampling parameters, and evaluation metrics. Appendix D.1 details the calibration configurations and statistical comparisons, while Appendix C establishes the population-coverage guarantee. The supplementary results report the evaluated configurations and their numerical outcomes. For the scheduling experiments, Appendix E specifies the request trace, arrival process, scheduling rules, and confidence-interval calculations, supporting reproduction of the simulation protocol and interpretation of the reported results.

REFERENCES

Ruicheng Ao, Gan Luo, David Simchi-Levi, and Xinshang Wang. Optimizing llm inference: Fluid-guided online scheduling with memory constraints, 2025.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Jiaming Cheng, Duong The Do, and Duong Tung Nguyen. Robust kv cache management for llm serving under output token length uncertainty, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui

Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Yiding Feng, Zonghan Yang, and Yuhao Zhang. Competitive non-clairvoyant kv-cache scheduling for llm inference, 2026.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. Efficient llm scheduling by learning to rank. volume 37, pp. 59006–59029, 2024.

Zhenghao Gan, Yichen Bao, Yifei Liu, Chen Chen, Quan Chen, and Minyi Guo. Sagesched: Efficient llm scheduling confronting demand uncertainty and hybridity, 2026.

Patrick Jaillet, Jiashuo Jiang, Konstantina Mellou, Marco Molinaro, Chara Podimata, and Zijie Zhou. Online scheduling for llm inference with kv cache constraints, 2025.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. s^3: Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems*, 36:18015–18027, 2023.

Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios Xydis, and Dimitrios Soudris. throttll'em: Predictive gpu throttling for energy efficient llm inference serving. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1363–1378. IEEE, 2025.

Jie Kong, Wei Wang, Jiehan Zhou, and Chen Yu. Llm-coopt: A co-design and optimization framework for efficient llm inference on heterogeneous platforms, 2026.

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 Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Yuhan Liu, Cong Xu, Qi Jia, Yihua Wang, Feiyu Chen, Liang Jin, Lu Liu, Yaqian Zhao, Yuting Ding, and Xiang Li. H-mas: Hierarchical multi-agent scheduling for multi-tenant llm serving. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39051–39071, 2026.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl, 2025.

Glen Messenger. Alps: Activation-based length prediction for intelligent llm inference scheduling. *Zenodo*, 2026. URL https://zenodo.org/records/19078431.

Sangdon Park, Osbert Bastani, Nikolai Matni, and Insup Lee. Pac confidence sets for deep neural networks via calibrated prediction. 2019.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation - A kvcache-centric architecture for serving LLM chatbot. In Haryadi S. Gunawi and Vasily Tarasov (eds.), *23rd USENIX Conference on File and Storage Technologies, FAST 2025, Santa Clara, CA, February 25-27, 2025*, pp. 155–170. USENIX Association, 2025. URL https://www.usenix.org/conference/fast25/presentation/qin.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. Efficient interactive llm serving with proxy model-based sequence length prediction, 2024.

Feiyang Ren, Shengtao Wen, Lingbing Guo, Yu Tian, Yuanning Cui, and Xiang Chen. When entropy is not enough: Reclaiming lost semantics in llm output length prediction, 2026.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. volume 32, 2019.

Rana Shahout, Eran Malach, Chunwei Liu, Weifan Jiang, Minlan Yu, and Michael Mitzenmacher. Don't stop me now: Embedding based scheduling for llms. In *International Conference on Learning Representations*, volume 2025, pp. 63345–63368, 2025.

Haiying Shen and Tanmoy Sen. Econoserve: Maximizing multi-resource utilization with slo guarantees in llm serving, 2024.

Haiying Shen, Tanmoy Sen, and Masahiro Tanaka. Mitigating kv cache competition to enhance user experience in llm inference, 2025.

Jing Wang, Yu-Yang Qian, Ke Xue, Chao Qian, Peng Zhao, and Zhi-Hua Zhou. Robust length prediction: A perspective from heavy-tailed prompt-conditioned distributions, 2026a.

Zhibin Wang, Zetao Hong, Xue Li, Zibo Wang, Shipeng Li, Qingkai Meng, Qing Wang, Chengying Huan, Rong Gu, Sheng Zhong, et al. Star: Decode-phase rescheduling for llm inference. In *Proceedings of the 2026 The 35th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 125–138, 2026b.

Samuel S Wilks. Determination of sample sizes for setting tolerance limits. *The Annals of Mathematical Statistics*, 12(1):91–96, 1941.

Huanyi Xie, Yubin Chen, Liangyu Wang, Lijie Hu, and Di Wang. Predicting llm output length via entropy-guided representations. 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Haoyu Zheng, Yongqiang Zhang, Fangcheng Fu, Xiaokai Zhou, Hao Luo, Hongchao Zhu, Yuanyuan Zhu, Hao Wang, Xiao Yan, and Jiawei Jiang. Scheduling llm inference with uncertainty-aware output length predictions, 2026.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. *Advances in Neural Information Processing Systems*, 36:65517–65530, 2023.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In Ada Gavrilovska and Douglas B. Terry (eds.), *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pp. 193–210. USENIX Association, 2024. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

A Related Work

Predicting response length. Length prediction has been used to estimate token counts, length buckets, and relative ranks for batching and shortest-job-first scheduling (Zheng et al., 2023; Jin et al., 2023; Qiu et al., 2024; Fu et al., 2024). ALPS, EGTP, and ESTP use target-model Prefill activations; TRAIL and STAR update remaining-length estimates during Decode (Messenger, 2026; Xie et al., 2026; Ren et al., 2026; Shahout et al., 2025; Wang et al., 2026b). RACE predicts a calibrated upper token budget before decoding.

Stochastic length and uncertainty. ProD, TIE, and SageSched study response-length variation for distributional prediction or request ordering (Wang et al., 2026a; Zheng et al., 2026; Gan et al., 2026). RACE learns location and scale from repeated responses, then calibrates the budget on independent prompts. It applies Wilks' classical one-sided tolerance rank (Wilks, 1941), complementing held-out calibration in conformal prediction and risk control (Romano et al., 2019; Park et al., 2019; Bates et al., 2021).

Prediction-aware serving. EconoServe, throttLL'eM, CacheOPT, and RobustKV use length predictions for KV-cache, routing, or energy decisions (Shen & Sen, 2024; Kakolyris et al., 2025; Shen et al., 2025; Cheng et al., 2026). H-MAS is a close precedent for upper-budget admission: it adds an empirical residual quantile within prompt-length buckets to a point prediction, then adjusts the token budget and concurrency (Liu et al., 2026). RACE learns a request-specific scale from Prefill features and calibrates the margin on independent prompts. Prediction-free methods also address KV admission (Jaillet et al., 2025; Ao et al., 2025; Feng et al., 2026).

B Method and Data Details

B.1 Feature Construction

This section specifies the feature configuration used in the primary Qwen3-8B experiment. Both predictors receive the same feature vector, constructed from the prompt length and the last prompt-token states at four model depths.

State selection and normalization. For a model with D decoder layers, we select the depths nearest $D/4$, $D/2$, $3D/4$, and D. In the primary Qwen3-8B configuration, $D = 36$ and the selected depths are

$$\mathcal{L} = \{\ell_1, \ell_2, \ell_3, \ell_4\} = \{9, 18, 27, 36\}.$$

Let r_{ℓ_j} denote the output of decoder block ℓ_j at the last prompt-token position, and let N denote the model's terminal RMSNorm. The states used for feature extraction are

$$h_j = \begin{cases} r_{\ell_j}, & \ell_j < D, \\ N(r_D), & \ell_j = D. \end{cases} \tag{9}$$

This follows the stored hidden-state convention: intermediate states are taken before terminal normalization, while the final-depth state includes that normalization.

Shared Gaussian projection. Each selected state is projected from the model hidden dimension d_{model} to $d_{\text{proj}} = 256$ dimensions:

$$z_j = G^\top h_j, \qquad G \in \mathbb{R}^{d_{\text{model}} \times d_{\text{proj}}}, \qquad G_{ab} \sim \mathcal{N}\left(0, \frac{1}{d_{\text{proj}}}\right). \tag{10}$$

The matrix is generated on the CPU in float32 using PyTorch seed 17. The same matrix is used at all four depths and remains fixed during training and inference. For Qwen3-8B, $d_{\text{model}} = 4096$, so G has shape 4096×256.

Scalar measurements. RACE supplements the projected states with the input token count, state norms, comparisons between successive selected depths, and scores for four selected tokens. Table 2 gives the complete feature-group composition.

Table 2: **Feature composition of the primary RACE input.** Counts refer to stored input columns. The scalar component contains 35 columns, including the fixed-zero metadata fields and duplicated norm entries described below.

Feature group	Definition	Count
Projected states	Four vectors $z_j = G^\top h_j$, each with 256 coordinates.	1,024
Input length	Number of input tokens after applying the prompt template.	1
Input-state metadata	Four generation and input-truncation fields, fixed to zero under the primary extraction protocol.	4
State norms	$\|h_j\|_2$ at each selected depth, stored in two columns per depth.	8
Cross-depth cosine similarity	c_j for the three successive selected-depth pairs.	3
Cross-depth distance	e_j for the same three pairs.	3
Selected-token scores	Four output-head scores at each of the four selected depths.	16
Total	Shared input to both predictors.	1,059

For successive selected depths, we compute

$$c_j = \frac{h_{j-1}^\top h_j}{\max(\|h_{j-1}\|_2, \epsilon)\max(\|h_j\|_2, \epsilon)}, \tag{11}$$

$$e_j = \|h_j - h_{j-1}\|_2, \qquad j \in \{2, 3, 4\}, \tag{12}$$

where $\epsilon = 10^{-8}$ matches the cosine-similarity implementation. These comparisons are between successive selected depths, such as layers 9 and 18, rather than between consecutive decoder blocks.

Selected-token scores. The selected tokens are EOS, `<think>`, `</think>`, and newline, corresponding to response termination, reasoning boundaries, and line breaks. Let w_t be the model's output-head weight vector for token t. At each selected depth, the score is

$$g_{j,t} = w_t^\top N(h_j). \tag{13}$$

These are pre-softmax scores computed with the output-layer weights. The terminal normalization is applied to every stored state h_j, including the final-depth state defined in Equation 9.

In the primary tokenizer, the IDs for EOS, `<think>`, `</think>`, and newline are 151645, 151667, 151668, and 198, respectively. Four scores at each of four depths contribute 16 scalar columns.

Stored input format. The four input-state metadata fields are `prediction_horizon`, `consumed_decode_tokens`, `decode_prefix_tokens_in_features`, and `was_truncated_to_max_input`. They are zero under the primary protocol, which extracts features before decoding and uses prompts without input truncation.

At each selected depth, the fields with suffixes `token_norm_last` and `last_norm` both store $\|h_j\|_2$. The input format retains these duplicate entries and the metadata fields to match the fitted predictors.

The total input dimension is therefore

$$4 \times 256 + \underbrace{(1 + 4 + 8 + 3 + 3 + 16)}_{\text{35 scalar columns}} = 1{,}059. \tag{14}$$

Thus, 1,059 is the dimension obtained by concatenating the specified feature groups. The projection width is the configured dimensionality parameter.

The resulting vector is denoted by $\phi(S)$. Fitting and inference use the same ordered feature names, selected depths, tokenizer IDs, and projection configuration. Extraction checks these quantities against the saved feature metadata.

B.2 Predictor Fitting

Training targets. Each fitting prompt contributes one pair of targets, defined in Equation 4. The targets use the lengths of naturally terminated responses; a prompt must have at least two such responses to contribute to fitting or validation.

Table 3: **Selected LightGBM configurations in the primary experiment.** The boosting-iteration counts are selected by validation and used for prediction.

Parameter	Length predictor	Variation predictor
Maximum leaves per tree	63	31
Minimum samples per leaf	40	20
Feature-sampling fraction	0.80	1.00
Selected boosting iterations	990	408
Random seed	42	1042

Quantiles are computed using NumPy's linear interpolation convention. When three naturally terminated responses are available, the difference between the 80th and 20th percentiles equals $0.6(\max O_i - \min O_i)$. The length-variation target has a floor of 32 tokens.

Optimization. Both predictors are LightGBM gradient-boosted tree regressors. The response-length predictor fits $\log(1 + m_i)$, and the length-variation predictor fits $\log(1 + d_i)$, using L1 loss.

Both use a learning rate of 0.025, at most 1,200 boosting rounds, row subsampling of 0.9 at every boosting round, and an L2 regularization coefficient of 1. No explicit maximum tree depth is imposed. Early stopping uses the validation loss in log space with a patience of 80 rounds.

We evaluate four hyperparameter configurations. Each tuple specifies the maximum number of leaves per tree, minimum samples per leaf, and feature-sampling fraction:

$$(15, 40, 0.65), \quad (31, 40, 0.65), \quad (31, 20, 1.00), \quad (63, 40, 0.80).$$

The response-length predictor is selected by validation mean absolute error in token units. The length-variation predictor is selected by validation mean absolute error in log space. Table 3 lists the selected primary configurations.

Returning predictions to token units. Let f_μ and f_s denote the fitted regressors, whose outputs are in log space. Their predictions are converted to token units and clipped as follows:

$$\mu(S) = \mathrm{clip}\big(\exp(f_\mu(\phi(S))) - 1,\ 0,\ H\big), \tag{15}$$

$$s(S) = \mathrm{clip}\big(\exp(f_s(\phi(S))) - 1,\ 32,\ H\big). \tag{16}$$

Both predictors are fixed before the independent calibration step described in Section 3.

B.3 Datasets and Splitting

Primary DeepScaleR cohort. The primary collection uses the `deepscaler.json` file from `agentica-org/DeepScaleR-Preview-Dataset` at revision `4500022`. For deduplication, each problem is assigned a key obtained by collapsing consecutive whitespace. This removes 1,218 duplicate entries from 40,315 records, leaving 39,097 unique problems. The original problem text is retained in the model input.

Unique problems are ordered by seeded SHA-256 hashes of their normalized text keys, using seed 20260816. The first 1,536 form the development pool, and the next 512 form the test set. The development pool is divided at the prompt level into 1,074 fitting, 185 validation, and 277 calibration prompts.

Each development prompt has three sampled responses, and each test prompt has two. Generation uses temperature 0.6, top-p 0.95, top-k 20, and a 32,768-token output limit. The collection limit and service limit are both $H = 32{,}768$ in this experiment.

After requiring at least two naturally terminated responses per prompt, 1,067 fitting prompts and 184 validation prompts remain. Calibration uses all 277 calibration prompts, with one prespecified response per prompt. Responses stopped at the collection limit receive the calibration target H.

Additional datasets. The ShareGPT experiments use the Aeala V4.3 cleaned-split source and maintain conversation-disjoint splits. All examples from a conversation remain in the same split. The

GSM8K experiments preserve the official training–test separation; fitting, validation, and calibration prompts are drawn from the training partition.

B.4 Software and Hardware

The primary experiment uses Python 3.12.12, PyTorch 2.8.0 with CUDA 12.8, Transformers 4.57.1, vLLM 0.11.0, LightGBM 4.6.0, NumPy 1.26.4, SciPy 1.16.3, pandas 2.3.3, and scikit-learn 1.7.2. Primary generation was sharded across four 48-GB NVIDIA RTX A6000 GPUs. The predictors were trained on an AMD EPYC 7502 host.

C Proof of the Population-Coverage Guarantee

Setup and guarantee. Fix $\alpha, \delta \in (0,1)$ and the fitted predictors μ and s, with $s(S) > 0$. Conditional on the data used to fit and select these predictors, assume that the n calibration requests and a future request are independent draws from the same population under the same decoding policy. Each calibration prompt contributes one prespecified sampled response.

Let $\mathcal{C}$ denote the calibration sample. The budget rule U_α depends on $\mathcal{C}$ through the calibrated multiplier q. We prove

$$\Pr_{\mathcal{C}}\big[\Pr\big\{Y^H \le U_\alpha(S) \,\big|\, \mathcal{C}\big\} \ge \alpha\big] \ge 1-\delta. \tag{17}$$

The inner probability is over an independent future prompt and its sampled response; the outer probability is over calibration samples.

Service-length and calibration scores. For calibration request i, define

$$A_i = \frac{Y_i^H - \mu_i}{s_i}, \qquad a_i = \frac{y_i^{\mathrm{up}} - \mu_i}{s_i}, \tag{18}$$

where $\mu_i = \mu(S_i)$ and $s_i = s(S_i)$. The target construction in Section 3 ensures $y_i^{\mathrm{up}} \ge Y_i^H$: a naturally terminated response provides its observed length, and a response stopped by the collection limit receives target H.

Since $s_i > 0$, the actual calibration scores satisfy

$$a_i \ge A_i \quad \text{for every } i, \qquad a_{(k)} \ge A_{(k)} \quad \text{for every } k. \tag{19}$$

The second inequality follows because increasing individual observations cannot decrease any of their order statistics.

Order-statistic coverage bound. Let F be the CDF of the service-length score

$$A = \frac{Y^H - \mu(S)}{s(S)}.$$

The scores $A_1, \dots, A_n$ are iid with this CDF. Define

$$t_\alpha = \inf\{t : F(t) \ge \alpha\}, \qquad p_< = \Pr\{A < t_\alpha\} = F(t_\alpha^-) \le \alpha.$$

The right-continuity of F gives $F(t_\alpha) \ge \alpha$.

Let

$$N_< = \sum_{i=1}^{n} \mathbf{1}\{A_i < t_\alpha\}.$$

Then $N_< \sim \mathrm{Bin}(n, p_<)$. If $N_< \le k-1$, the kth smallest score satisfies $A_{(k)} \ge t_\alpha$, and hence $F(A_{(k)}) \ge \alpha$. Therefore,

$$\Pr_{\mathcal{C}}\{F(A_{(k)}) \ge \alpha\} \ge \Pr\{\mathrm{Bin}(n, p_<) \le k-1\} \tag{20}$$

$$\ge \Pr\{\mathrm{Bin}(n, \alpha) \le k-1\} \tag{21}$$

$$= \Pr\{Z \ge \alpha\}, \qquad Z \sim \mathrm{Beta}(k, n+1-k). \tag{22}$$

The second inequality follows from $p_< \le \alpha$, and the last equality is the beta–binomial identity. For a rank satisfying Equation 6, the final probability is at least $1-\delta$. Counting scores strictly below t_α makes this argument valid for discrete scores and tied values.

From the selected score to the budget. For a feasible rank k, RACE uses

$$q = \max\{a_{(k)}, 0\} \geq a_{(k)} \geq A_{(k)}.$$

For a future request, the event $A \leq q$ implies

$$Y^H \leq \mu(S) + q\, s(S).$$

Because $0 \leq Y^H \leq H$, this coverage event is preserved when the right-hand side is clipped to $[0, H]$. Consequently, for a fixed calibration sample,

$$\Pr\{Y^H \leq U_\alpha(S) \mid \mathcal{C}\} \geq F(q) \tag{23}$$

$$\geq F(A_{(k)}). \tag{24}$$

Combining Equations 22 and 24 proves Equation 17.

If no rank satisfies Equation 6, RACE returns $U_\alpha(S) = H$. This fallback covers the service response length for every request, so the guarantee also holds in this case.

Calibration sample size. For fixed n and α, the rank criterion becomes easier to satisfy as k increases. At the largest available rank, $k = n$, its confidence bound is

$$\Pr\{Z \geq \alpha\} = 1 - \alpha^n, \qquad Z \sim \mathrm{Beta}(n, 1).$$

A feasible rank therefore exists if and only if $1 - \alpha^n \geq 1 - \delta$, equivalently,

$$n \geq \left\lceil \frac{\log \delta}{\log \alpha} \right\rceil. \tag{25}$$

This condition determines when the order-statistic rule can select a calibration score. The resulting reservation size depends on the observed scores, the fitted predictors, and the service limit.

D Extended Results

D.1 Evaluation Protocol and Paired Comparisons

Calibration and baselines. The primary split contains 1,074 fitting, 185 validation, and 277 calibration prompts. Requiring two naturally terminated responses leaves 1,067 fitting and 184 validation prompts. Calibration retains all 277 prompts and uses one prespecified response per prompt, selecting rank 270 for 95% coverage at 95% confidence. The test set contains 19 capped responses among 1,024 outputs. Service coverage uses lengths capped at H; the natural-length coverage lower bound counts unresolved responses as misses.

Within each setting, calibrated methods share the data split, service limit, and coverage and confidence targets. The calibration sample sizes and ranks are 185/181 for Qwen3-30B-A3B/GSM8K, 93/92 for DeepSeek-V2-Lite/DeepScaleR, and 357/347 for fresh ShareGPT. The ShareGPT GBM predictor was trained with a 64K collection limit and recalibrated for the 32K service limit; RACE, GBM, and Global configurations were fixed before test generation. DeepSeek predictors were fixed before testing, and its paired analysis was specified afterward.

AFT-PAC uses XGBoost accelerated-failure-time trees with logistic noise on $\log(1 + L)$. Repeated responses receive equal total weight per prompt, and capped responses enter as right-censored intervals. It uses the grouped GBM baseline's final-token prefill feature mask and is fixed before independent calibration.

Paired comparisons. We compute two-sided 95% percentile intervals from paired prompt-level bootstrap samples, retaining both test responses from each sampled prompt. We use 20,000 replicates, except for Qwen3-30B-A3B/GSM8K, which uses 5,000. Coverage is noninferior to GBM-PAC when the interval's lower endpoint for the coverage difference is at least -1 percentage point. Table 4 reports these comparisons; cross-setting intervals are descriptive and have no multiplicity adjustment.

Table 4: **Paired comparisons with GBM-PAC.** Coverage differences are percentage points; budget savings are percentages. Cov. is service coverage, and U/H is mean budget as a percentage of the service limit. NI indicates whether the coverage noninferiority criterion passes. Brackets contain 95% intervals.

Model / task	Evidence	RACE		Δcov.	U saving	NI
		Cov.	U/H	vs. GBM [95% CI]	vs. GBM [95% CI]	
Qwen3-8B / DeepScaleR	prospective	97.56	66.3	+0.59 [-0.68,+1.86]	9.94 [7.89,12.07]	pass
DeepSeek-V2-Lite / DeepScaleR	pretest frozen	98.44	12.5	+0.20 [-0.39,+0.98]	22.26 [18.41,25.97]	pass
Qwen3-8B / ShareGPT	prospective	96.48	12.8	-0.98 [-2.25,+0.20]	20.55 [18.24,22.84]	fail
Qwen3-30B-A3B / GSM8K	prospective	96.47	9.1	-0.72 [-1.44,+0.00]	11.87 [10.25,13.42]	fail

Additional test prompts. The additional cohort contains 500 DeepScaleR prompts, excluding the original development and test prompts and their near-duplicates. Four responses per prompt give 2,000 outputs under the primary sampling policy and 32,768-token limit. RACE, GBM-PAC, and matched-input Mixture-PAC share 1,059 prefill features; native-input Mixture-PAC adds richer boundary statistics. All use the original 277 calibration prompts. Coverage-target experiments recalibrate fixed predictors; margin experiments refit the variation predictor while keeping the length predictor fixed.

Table colors. In Table 1, budget saving is $100(1 - \bar{U}/\bar{U}_{\mathrm{GBM}})$. Blue and orange indicate smaller and larger budgets than GBM-PAC, with shading saturated at a 25% change. Coverage is unshaded; statistical comparisons are reported separately above.

Reservation components. Table 5 supplements the primary results with excess and deficit reservations and a variant trained on the full within-prompt length range.

Table 5: **Primary reservation components.** All methods use the same 512 prompts and 1,024 outputs. Over and Under are mean excess and deficit tokens over naturally terminated outputs, including zero values. The range variant replaces the interquantile target with the maximum minus minimum naturally terminated response length, retaining the 32-token floor.

Method	Cov. (%) ↑	Mean U ↓	Over ↓	Under ↓
RACE	97.56	21,721	12,558	61
RACE (range scale)	98.34	22,117	12,933	47
GBM-Quantile-PAC	96.97	24,118	15,018	42
Mixture-PAC	97.85	21,552	12,377	64
Global-PAC	96.48	28,367	19,363	37
Service limit	100.00	32,768	23,727	0

D.2 Safety Margin Allocation

We fix the length predictor and the same 1,051 fitting, 183 validation, and 277 calibration prompts. The variation predictor is trained on nested sets of $n_{\mathrm{resp}} \in \{2, 3, 5, 8\}$ responses per prompt and compared with constant, shuffled, and length-proportional alternatives. Each rule is calibrated separately and evaluated on the same 2,000 outputs. Table 6 provides the complete values for the main-text margin experiment.

D.3 Training with a Fixed Sampling Budget

We also compare response counts using all eligible prompts at each count and under three fixed budgets for generating variation labels. Together with the fixed-cohort comparison above, these experiments contain 80 separately calibrated conditions on the same 2,000 test outputs. The new collection adds 6,295 responses and 58,795,260 generated tokens, including 89 capped responses, to 4,608 archived responses. Generation cost includes capped responses before eligibility filtering; the shared length-predictor training cost is recorded separately.

Table 6: **Safety margin allocation at fixed training prompts.** n_{resp} is the number of responses per training prompt. Budget is mean reserved output tokens; coverage is the percentage of test outputs within budget. All rows use 1,051 fitting prompts.

Margin rule	n_{resp}	Mean budget	Coverage (%)
Learned (RACE)	2	21233.47	96.50
Constant	2	25431.25	97.60
Shuffled	2	23325.26	95.70
Length-proportional	2	22464.94	97.50
Learned (RACE)	3	21542.40	96.90
Constant	3	25431.25	97.60
Shuffled	3	24816.22	95.95
Length-proportional	3	22464.94	97.50
Learned (RACE)	5	21250.51	96.40
Constant	5	25431.25	97.60
Shuffled	5	23842.90	95.10
Length-proportional	5	22464.94	97.50
Learned (RACE)	8	21828.89	96.80
Constant	8	25431.25	97.60
Shuffled	8	25122.70	96.25
Length-proportional	8	22464.94	97.50

The three budgets are 5,925,799, 11,851,599, and 23,703,199 tokens, equal to 25%, 50%, and 100% of the reference two-response label cost. Within a fixed budget, sampling more responses per prompt leaves fewer distinct prompts for fitting. Figures 6 and 7 show the complete comparisons; Tables 7 and 8 give their values.

Table 7: **Results using all eligible prompts at each response count.** Budget and coverage use the same definitions and test outputs as Table 6; fitting-prompt counts vary with the availability of naturally terminated responses.

Margin rule	n_{resp}	Fit prompts	Mean budget	Coverage (%)
Learned (RACE)	2	1051	21233.47	96.50
Constant	2	1051	25431.25	97.60
Shuffled	2	1051	23325.26	95.70
Length-proportional	2	1051	22464.94	97.50
Learned (RACE)	3	1067	21229.17	96.40
Constant	3	1067	25431.25	97.60
Shuffled	3	1067	24186.42	95.30
Length-proportional	3	1067	22464.94	97.50
Learned (RACE)	5	1071	21599.69	96.85
Constant	5	1071	25431.25	97.60
Shuffled	5	1071	24280.32	95.20
Length-proportional	5	1071	22464.94	97.50
Learned (RACE)	8	1072	21534.27	96.45
Constant	8	1072	25431.25	97.60
Shuffled	8	1072	25425.39	96.30
Length-proportional	8	1072	22464.94	97.50

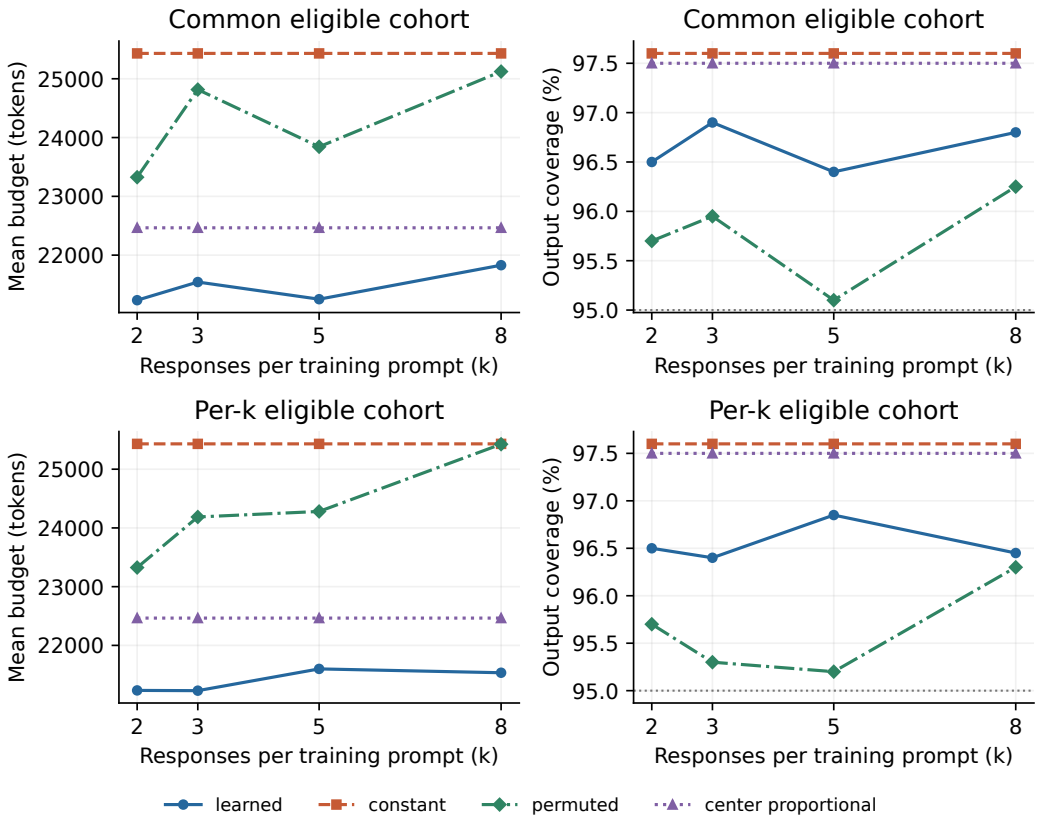

Figure 6: **Response counts with fixed and available training cohorts.** The upper row fixes the eligible prompts across response counts; the lower row uses all eligible prompts at each count. The 32 conditions share the length predictor, calibration prompts, and test outputs.

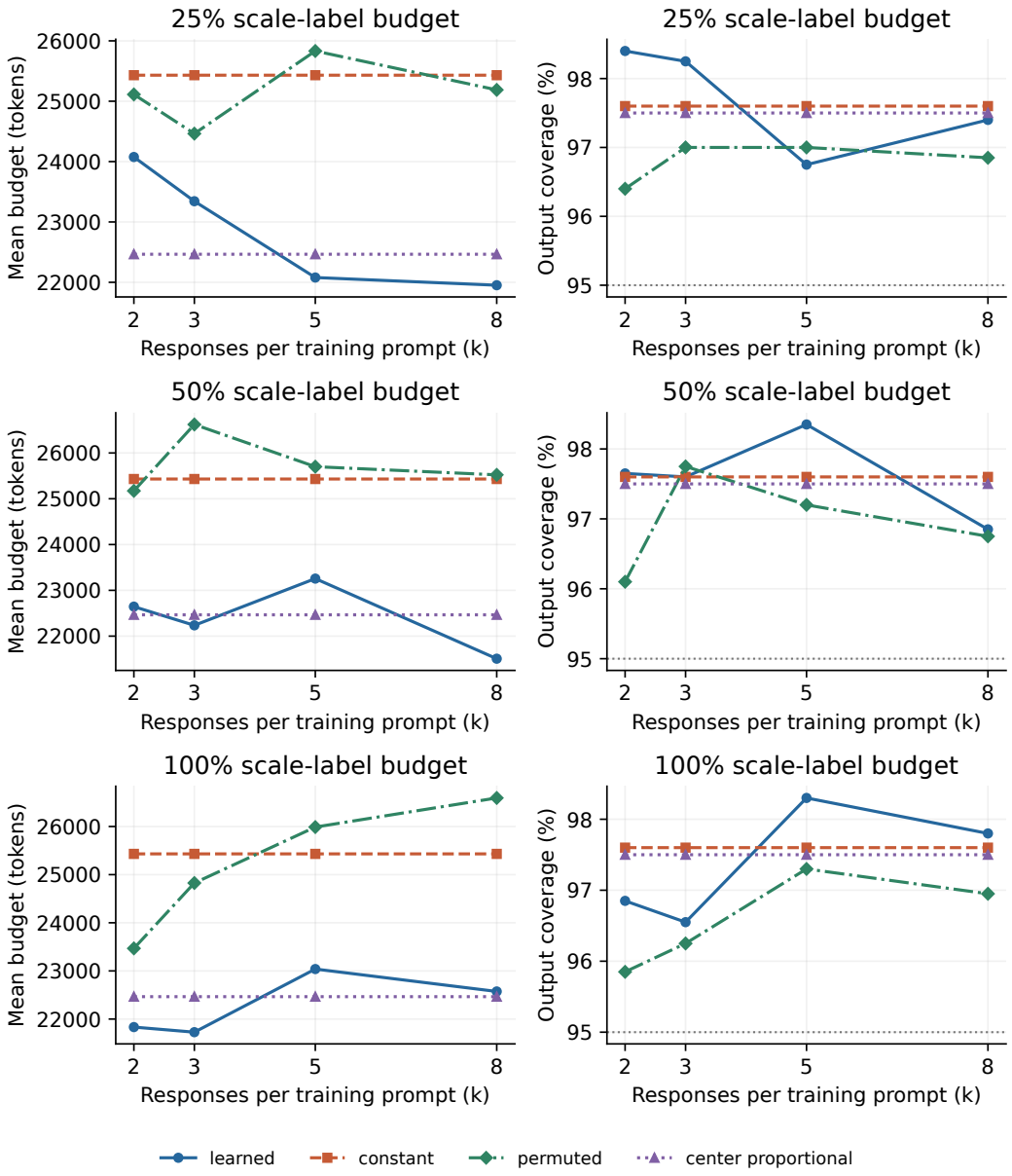

Figure 7: **Safety margins under fixed label-generation budgets.** Rows show the 25%, 50%, and 100% budgets, covering 48 conditions. The length predictor, calibration prompts, and test outputs are shared. Table 8 includes the corresponding fitting-prompt counts.

Table 8: **All fixed-budget sampling comparisons.** Each budget is charged before filtering capped responses. Fit is the number of eligible fitting prompts, Budget is mean reserved output tokens, and Cov. is service coverage (%). All conditions use the same length predictor, calibration prompts, and test outputs.

		25% label budget			50% label budget			100% label budget		
Margin rule	n_{resp}	Fit	Budget	Cov.	Fit	Budget	Cov.	Fit	Budget	Cov.
Learned (RACE)	2	258	24075.98	98.40	516	22644.00	97.65	1051	21833.01	96.85
Constant	2	258	25431.25	97.60	516	25431.25	97.60	1051	25431.25	97.60
Shuffled	2	258	25112.52	96.40	516	25172.10	96.10	1051	23467.91	95.85
Length-proportional	2	258	22464.94	97.50	516	22464.94	97.50	1051	22464.94	97.50
Learned (RACE)	3	185	23343.29	98.25	354	22235.43	97.60	693	21727.34	96.55
Constant	3	185	25431.25	97.60	354	25431.25	97.60	693	25431.25	97.60
Shuffled	3	185	24463.09	97.00	354	26622.15	97.75	693	24826.19	96.25
Length-proportional	3	185	22464.94	97.50	354	22464.94	97.50	693	22464.94	97.50
Learned (RACE)	5	116	22079.35	96.75	218	23256.20	98.35	432	23038.63	98.30
Constant	5	116	25431.25	97.60	218	25431.25	97.60	432	25431.25	97.60
Shuffled	5	116	25832.10	97.00	218	25700.50	97.20	432	25988.35	97.30
Length-proportional	5	116	22464.94	97.50	218	22464.94	97.50	432	22464.94	97.50
Learned (RACE)	8	72	21951.68	97.40	143	21506.92	96.85	263	22573.10	97.80
Constant	8	72	25431.25	97.60	143	25431.25	97.60	263	25431.25	97.60
Shuffled	8	72	25188.07	96.85	143	25522.28	96.75	263	26593.06	96.95
Length-proportional	8	72	22464.94	97.50	143	22464.94	97.50	263	22464.94	97.50

D.4 Feature and Margin Ablations

These post-test checks use the original 512 prompts and retain the primary RACE configuration as the reference. Table 9a varies the prediction features. Four projected states reduce the mean reservation from 29,347 tokens with prompt-length-only features to 21,924 tokens, with observed coverage of 98.44% and 98.05%, respectively. The final-state variant also produces reservations close to those of RACE.

Panel (b) fixes the length predictor and varies the safety-margin rule. Shuffling the learned variation increases the mean reservation from 21,721 to 25,983 tokens at the same 97.56% coverage. Relative to constant margins, RACE saves 14.88% with a coverage difference of -0.29 percentage points (95% interval $[-1.37, +0.78]$); relative to length-proportional margins, it saves 4.10% with coverage 1.07 points lower.

D.5 Choice of Calibration Response

We keep the fitted predictors, 277 calibration prompts, and test outputs fixed, but replace each prompt's prespecified calibration response with one of its two other sampled responses. Table 10 shows the resulting variation in budget and coverage.

D.6 Model and Dataset Settings

Table 11 separates the four main settings from earlier development, boundary, and transfer evaluations. Service coverage measures the capped output; Natural LB counts unresolved natural lengths as misses.

D.7 Response-Length Variation by Predicted Group

We divide the 512 primary test prompts into four equal groups by predicted variation $s(S)$. Length differences and prediction errors use the 496 prompts with two naturally terminated responses. Across these prompts, $s(S)$ correlates with the absolute response-length difference (Spearman $\rho =$

Table 9: **Feature and margin ablations on the primary test set.** (a) Input features vary under the same 95% coverage and 95% confidence calibration rule. MAE evaluates the response-length predictor. (b) The length predictor is fixed while the variation rule changes. Over and Under follow Table 5; Spearman correlations use 1,005 naturally terminated responses.

(a) Feature ablation

Probe	Dim.	Center MAE	Cov. (%)	Mean U
Prompt length	1	4,055	98.44	29,347
Boundary scalars	40	3,225	97.36	22,603
Four residuals	1024	3,010	98.05	21,924
Final residual + prompt	265	3,065	98.05	21,744
RACE (full feature set)	1064	2,995	98.14	21,966
RACE	1059	3,044	97.56	21,721

(b) Safety-margin controls

Variation rule	Svc. cov. (%)	Mean U	Over	Under	$\rho(s, \lvert L - \mu \rvert)$
RACE (learned scale)	97.56	21,721	12,558	61	0.398
Constant / additive PAC	97.85	25,518	16,387	16	–
Center proportional	98.63	22,650	13,473	45	0.417
Permuted learned scale	97.56	25,983	16,942	86	0.004

Table 10: **Changing the response selected for calibration.** Response 0 is the prespecified choice. Cap hits counts responses reaching the collection limit; Adjustment is the selected score $a_{(k)}$. Natural LB counts unresolved natural lengths as misses; U/H is expressed as a percentage.

Response choice	Cap hits	Adjustment	Svc. cov. (%)	Natural LB (%)	Mean U	U/H (%)
0	3	10.346	97.56	96.29	21,721	66.3
1	2	10.465	97.66	96.39	21,830	66.6
2	4	10.937	98.24	96.88	22,254	67.9

Table 11: **Main and additional evaluation settings.** The first four rows correspond to the main comparison. Evidence roles distinguish prospective (pros.) and retrospective (retro.) evaluations. Prompts$\times R$ gives prompts and responses per prompt. Coverage and U/H are percentages; H and mean U are tokens.

Model / task	Evidence role	Prompts$\times R$	H	Svc. cov.	Nat. LB	Mean U	U/H
Qwen3-8B / DeepScaleR	pros. primary	512×2	32K	97.56	96.29	21,721	66.3
Qwen3-8B / ShareGPT-fresh	pros. repl.	512×2	32K	96.48	96.48	4,208	12.8
Qwen3-30B-A3B / GSM8K	pros. MoE	1,319×2	32K	96.47	96.47	2,976	9.1
DeepSeek-V2-Lite / DeepScaleR	pretest; stat. post	256×2	64K	98.44	98.44	8,209	12.5
Qwen3-8B / ShareGPT	retro. dev.	984×2	32K	95.83	95.83	4,399	13.4
R1-Distill-Qwen-1.5B / DeepScaleR	retro. scale bdry.	256×2	64K	98.44	94.92	51,977	79.3
R1-Distill-Qwen-7B / DeepScaleR	retro. boundary	256×2	64K	99.61	99.22	52,255	79.7
Qwen3-8B / HumanEval (59-cal)	retro. target-cal	105×2	32K	99.52	99.52	17,325	52.9
Qwen3-8B / MBPP→HumanEval	negative transfer	164×2	32K	99.70	99.39	30,774	93.9

0.411, 95% interval $[0.334, 0.485]$) and mean absolute length-prediction error (0.477, $[0.402, 0.546]$). Table 12 gives the group-level values underlying the main-text analysis.

Table 12: **Response-length variation across predicted groups.** Q1 to Q4 run from lowest to highest predicted variation. Natural pairs counts prompts with two naturally terminated outputs. Pair < 1000 is the percentage of these pairs differing by fewer than 1,000 tokens. Mean point error averages the two absolute length-prediction errors. Budget and service coverage include all requests in each group.

Scale quartile	Natural pairs	Mean s	Mean $\lvert L_0 - L_1 \rvert$	Mean point error	Pair < 1000 (%)	Mean U	Svc. cov. (%)
Q1	128	648	1,017	1,834	71.9	10,801	96.09
Q2	127	1,083	1,575	2,355	45.7	18,055	97.66
Q3	123	1,578	2,239	3,440	35.8	25,670	96.48
Q4	118	2,472	3,406	4,591	28.0	32,359	100.00

D.8 Stratified Calibration and Output Limits

Stratified calibration. RACE-S divides ShareGPT requests at the median predicted variation fixed on validation data and calibrates each group at confidence $1 - \delta/2$ using a Bonferroni allocation. This prespecified secondary variant increases coverage from 96.48% to 97.27% and the mean reservation from 4,208 to 4,560 tokens (Table 13).

Table 13: **Stratified calibration on conversation-disjoint ShareGPT.** Low and High are the two groups' coverage rates. ΔU is the mean budget difference from RACE, with its 95% interval where available. Service coverage and Natural LB are percentages.

Method	Role	Svc. cov.	Nat. LB	Mean U	ΔU vs. RACE	Low cov.	High cov.
RACE	primary	96.48	96.48	4,208	–	–	–
RACE-S	prespec. secondary	97.27	97.27	4,560	+352 [+330, +375]	98.73	96.01
GBM-Quantile-PAC	baseline	97.46	97.46	5,296	+1,088	–	–
Global-PAC	baseline	97.75	97.75	7,055	+2,847	–	–
Hard service bound	ceiling	100.00	99.32	32,768	+28,560	–	–

Extending capped outputs. We extend every capped primary response from 32K to 40K while keeping the service limit at $H = 32{,}768$. Service coverage remains 97.56%; the natural-length coverage lower bound is 96.29%, with 0.586% of natural lengths still unresolved (Table 14).

Table 14: **Service and natural lengths after extending capped outputs to 40K.** Cov. evaluates the reserved budget; unresolved natural lengths count as misses. Cens. is the percentage with unresolved target lengths. MAE uses observed target lengths. Each $< \tau$ column is the percentage of the full cohort with an observed target and absolute length-prediction error below τ. Budgets, MAE, and thresholds are in tokens.

Estimand	Cov. [95% CI]	Cens. (%)	Mean U	MAE obs.	< 100	< 200	< 500	< 1000
Service $\min(L, 32\text{K})$	97.56 [96.39, 98.63]	0.000	21,721	3,401	2.4	4.9	12.1	26.7
Natural after 40K extension	96.29 [94.92, 97.56]	0.586	21,721	3,345	2.4	4.9	12.1	26.7

D.9 Budgets at Different Coverage Targets

We recalibrate four fixed predictors at coverage targets of 90%, 92.5%, 95%, 97.5%, and 99%, with confidence fixed at 95%. The 277 calibration prompts and 2,000 test outputs are shared. Only calibration changes; GBM-PAC and Mixture-PAC retain their fitted 95th- and 50th-percentile functions, respectively. This sweep was specified after the initial confirmation comparison, whose primary target remains 95%.

The selected ranks for the first four targets are 258, 264, 270, and 275. At 99%, the largest rank gives confidence 0.9382, below 0.95, so all methods return $H = 32{,}768$. Their 100% service coverage at that target follows from this full-budget fallback. Matched-input Mixture-PAC has identical budgets at 90% and 92.5% because its negative calibration multipliers are clipped to zero.

Tables 15 and 16 give the complete values behind the main-text coverage analysis and the additional 99% target. Intervals use 5,000 paired resamples of the 500 test prompts, retaining all four responses per prompt. They are pointwise descriptive intervals conditional on the fitted predictors and calibration sample.

Table 15: **Budgets and coverage at all five targets.** Budgets are reserved output tokens; coverage is the percentage of service outputs within budget. All 99% targets use the full-budget fallback. Intervals use the prompt-level bootstrap described above.

Method	Target (%)	Mean budget	95% interval	Cov. (%)	95% interval
RACE	90	17,912.76	[17,213.36, 18,594.97]	92.50	[90.55, 94.35]
RACE	92.5	19,493.78	[18,768.83, 20,210.84]	94.40	[92.70, 95.95]
RACE	95	21,229.17	[20,487.21, 21,954.48]	96.40	[95.10, 97.55]
RACE	97.5	25,390.63	[24,707.22, 26,066.29]	99.55	[99.20, 99.85]
RACE	99	32,768.00	[32,768.00, 32,768.00]	100.00	[100.00, 100.00]
GBM-PAC	90	18,520.52	[18,162.99, 18,869.17]	92.90	[91.00, 94.60]
GBM-PAC	92.5	20,110.84	[19,747.31, 20,470.09]	93.75	[91.95, 95.35]
GBM-PAC	95	24,009.59	[23,634.55, 24,385.13]	96.90	[95.65, 97.95]
GBM-PAC	97.5	27,038.52	[26,693.53, 27,386.48]	98.25	[97.40, 99.00]
GBM-PAC	99	32,768.00	[32,768.00, 32,768.00]	100.00	[100.00, 100.00]
Mixture-PAC / matched	90	19,999.70	[19,238.34, 20,749.04]	92.75	[90.80, 94.55]
Mixture-PAC / matched	92.5	19,999.70	[19,238.34, 20,749.04]	92.75	[90.80, 94.55]
Mixture-PAC / matched	95	20,941.17	[20,172.20, 21,691.37]	95.25	[93.65, 96.70]
Mixture-PAC / matched	97.5	27,384.90	[26,774.40, 27,977.40]	99.85	[99.65, 100.00]
Mixture-PAC / matched	99	32,768.00	[32,768.00, 32,768.00]	100.00	[100.00, 100.00]
Mixture-PAC / native	90	19,416.64	[18,710.97, 20,127.18]	92.80	[90.90, 94.60]
Mixture-PAC / native	92.5	20,527.97	[19,796.24, 21,254.91]	94.90	[93.30, 96.35]
Mixture-PAC / native	95	21,488.27	[20,743.02, 22,232.23]	96.40	[95.10, 97.60]
Mixture-PAC / native	97.5	27,065.26	[26,442.28, 27,683.12]	99.75	[99.40, 100.00]
Mixture-PAC / native	99	32,768.00	[32,768.00, 32,768.00]	100.00	[100.00, 100.00]

Table 16: **Excess reservation and shortfall at the same targets.** Excess is $\max(U - Y^H, 0)$ and shortfall is $\max(Y^H - U, 0)$, averaged over all service outputs, including zero values, in tokens. Intervals use the same resamples as Table 15.

Method	Target (%)	Mean excess	95% interval	Mean shortfall	95% interval
RACE	90	8,737.70	[8,248.11, 9,220.44]	307.24	[217.80, 411.39]
RACE	92.5	10,213.19	[9,678.82, 10,733.95]	201.71	[133.40, 279.26]
RACE	95	11,863.05	[11,291.46, 12,417.36]	116.18	[69.39, 171.62]
RACE	97.5	15,925.42	[15,352.52, 16,494.96]	17.10	[1.76, 41.79]
RACE	99	23,285.70	[22,721.79, 23,824.08]	0.00	[0.00, 0.00]
GBM-PAC	90	9,421.99	[9,048.77, 9,775.16]	383.77	[280.00, 508.02]
GBM-PAC	92.5	10,911.13	[10,509.97, 11,289.60]	282.59	[199.83, 383.15]
GBM-PAC	95	14,640.84	[14,194.48, 15,079.65]	113.56	[68.89, 169.18]
GBM-PAC	97.5	17,604.72	[17,129.74, 18,071.84]	48.50	[22.21, 83.49]
GBM-PAC	99	23,285.70	[22,721.79, 23,824.08]	0.00	[0.00, 0.00]
Mixture-PAC / matched	90	10,724.57	[10,153.94, 11,285.10]	207.17	[136.36, 291.91]
Mixture-PAC / matched	92.5	10,724.57	[10,153.94, 11,285.10]	207.17	[136.36, 291.91]
Mixture-PAC / matched	95	11,618.33	[11,021.90, 12,206.78]	159.46	[100.16, 232.22]
Mixture-PAC / matched	97.5	17,912.63	[17,339.64, 18,492.17]	10.03	[0.00, 28.93]
Mixture-PAC / matched	99	23,285.70	[22,721.79, 23,824.08]	0.00	[0.00, 0.00]
Mixture-PAC / native	90	10,159.07	[9,656.33, 10,674.75]	224.73	[149.46, 313.62]
Mixture-PAC / native	92.5	11,202.44	[10,666.76, 11,746.29]	156.77	[95.94, 230.92]
Mixture-PAC / native	95	12,118.57	[11,561.93, 12,677.43]	112.60	[61.90, 177.05]
Mixture-PAC / native	97.5	17,597.30	[17,027.56, 18,163.64]	14.34	[0.00, 38.55]
Mixture-PAC / native	99	23,285.70	[22,721.79, 23,824.08]	0.00	[0.00, 0.00]

Table 17: **Scheduling effects on the 512-request trace.** Throughput and JCT changes are relative percentages for RACE versus the comparator; excursion changes are percentage points and peak-demand changes are tokens. "All 25" and "40K" use the scopes defined above. Brackets give 95% arrival-seed bootstrap intervals. Timing is in simulation steps.

Comparator	Scope	Δ Tok./step (%)	Δ Mean JCT (%)	Δ P95 JCT (%)
GBM-PAC	All 25	+4.01 [+3.86, +4.17]	-21.44 [-22.10, -20.79]	-7.35 [-7.76, -6.92]
GBM-PAC	40K	+16.12 [+15.83, +16.37]	-32.35 [-33.52, -30.98]	-18.95 [-19.53, -18.33]
Mixture-PAC	All 25	-1.45 [-1.57, -1.34]	+2.01 [+1.21, +2.86]	+2.22 [+1.84, +2.64]
Mixture-PAC	40K	-0.63 [-0.95, -0.32]	+1.04 [-0.85, +3.13]	+0.77 [+0.26, +1.33]

Comparator	Scope	Δ Excursion (pp)	Δ Peak physical KV (tokens)
GBM-PAC	All 25	+0.017 [+0.009, +0.023]	+989 [-521, +2444]
GBM-PAC	40K	+0.084 [+0.064, +0.102]	+9469 [+7602, +11118]
Mixture-PAC	All 25	+0.004 [-0.000, +0.008]	-390 [-1611, +601]
Mixture-PAC	40K	+0.062 [+0.045, +0.077]	+4242 [+3126, +5195]

E Scheduling Simulation and Reproducibility

E.1 Simulation Settings and Request Arrivals

We replay 512 requests with recorded prompt lengths, output lengths, and per-method reservation budgets. All methods receive identical request orders, arrival times, and output lengths. The simulator allows eight concurrent decoding requests and uses $H = 32{,}768$ output tokens. We combine five cache capacities, $K \in \{40{,}000, 48{,}000, 65{,}536, 98{,}304, 131{,}072\}$ token slots, with five loads, $\rho \in \{0.50, 0.75, 0.90, 1.05, 1.25\}$. Each combination uses ten arrival seeds: $11, 23, 37, 53, 71, 89, 107, 131, 163, 197$.

Each seed determines a request permutation and exponential inter-arrival times with mean $1/\lambda$, where

$$\lambda = \frac{S_{\max}\rho}{\overline{y}} = \frac{8\rho}{\overline{y}}. \tag{26}$$

Here $\overline{y}$ is mean output length, and ρ measures incoming token demand relative to nominal decoding capacity. Cumulative arrival times are rounded down to integer steps and shifted to start at zero.

E.2 Scheduling Rules and Metrics

The scheduler scans the queue in arrival order and admits the first request whose reservation satisfies Equation 8, repeating until no further request fits or eight requests are active. Reservations from Equation 3 remain fixed until completion. Each active request generates one token per step until reaching its recorded output length. The one-token minimum reservation and empty-system oversize fallback are never triggered by this trace.

Throughput is total output tokens divided by simulation duration. Job completion time (JCT) runs from arrival to completion and includes queueing; we report its mean and 95th percentile. Queueing delay runs from arrival to the start of decoding. All times are measured in simulation steps.

After generating $G_i(t)$ tokens, request i occupies $P_i + G_i(t)$ token slots. Requests continue when they exceed their budgets. A capacity excursion is a step when total demand exceeds K; its fraction is the number of such steps divided by simulation duration. Peak demand is the maximum combined occupancy. Byte summaries use 147,456 KV-cache bytes per token. The implementation advances between arrivals and completions, accumulating demand over the intervening steps.

E.3 Paired Comparisons and Confidence Intervals

Each RACE run is paired with the baseline run at the same capacity, load, and seed. Throughput and JCT changes are $100(v_{\mathrm{RACE}}/v_{\mathrm{baseline}} - 1)$; excursion changes are percentage-point differences, and peak-demand changes are token differences. All pairs receive equal weight. "All 25" averages all settings and ten seeds; "40K" averages five loads and ten seeds at $K = 40{,}000$.

We use 20,000 bootstrap replicates with analysis seed 20260818. Each replicate resamples the ten arrival seeds, retaining all settings from each selected seed. Two-sided 95% percentile intervals therefore measure arrival-sequence variation for the fixed request trace and configuration grid.

E.4 Scheduling Results

Table 17 provides the complete comparisons for the main-text scheduling analysis, including completion-time tails and cache-demand changes.