FASTER BLOCK-DIFFUSION SERVING WITH DISTRIBUTION-FREE RISK GUARANTEES

Anonymous authors
Paper under double-blind review

ABSTRACT

Block-diffusion language models are served at hand-picked operating points, such as acceptance thresholds, buffer depth, schedule, checkpoint and precision, and each point is chosen by its mean benchmark accuracy. However, a mean does not tell an operator how often a faster configuration fails on prompts that the slower one answers correctly. On the serving engine and its decode traces, the default commit rule already commits every fully resolved block, a static skip rule captures nearly all of the compute that allocation can save, and self-distillation on engine-decoded targets adds speed at unchanged accuracy. Larger speedups come from lower thresholds, which commit tokens that are still uncertain. We therefore present Redline, a finite-sample procedure that selects operating points, hand-picked or learned, from the correctness of their answers on calibration prompts. Redline keeps the reference-relative risk, the joint probability that the reference answers correctly and a candidate configuration does not, within a user-chosen budget with high probability, and deploys the fastest configuration that passes. It speeds up math at a smaller risk budget than code in both model families, and at a budget of ten percent it deploys a LLaDA2 math configuration that commits over a third more tokens in each forward. It also applies without modification to the acceptance rule of speculative decoding and to weight quantization. On the same calibration data, Redline stays within its stated failure probability, whereas each tolerance of a mean-accuracy rule either gains less speed for some model and task or exceeds the risk budget far more often for another. Code will be released upon acceptance.

1 INTRODUCTION

Block-diffusion language models (BD-LMs) generate text block by block. Each block is produced autoregressively and cached like an autoregressive prefix, while the tokens inside the block are denoised in parallel (Arriola et al., 2025; Cheng et al., 2026; Bie et al., 2025). Multi-block engines go further and keep several blocks in flight, which raises the number of tokens committed in each model forward (TPF) at equal or better accuracy (Jin et al., 2026). In practice, each such system is released with hand-picked serving settings, and its published operating points are chosen by mean benchmark accuracy (Bie et al., 2025; 2026; Cheng et al., 2026). A mean does not tell an operator how often, and with what confidence, a faster configuration fails on a prompt that the slower one answers correctly, because the prompts it newly gets right offset those it newly gets wrong.

The default response of the field is to *learn* a better frontier (Chen et al., 2026a; Qian et al., 2026; Wang et al., 2026; Hu et al., 2026; Lee et al., 2026b; Chen et al., 2026b; Bie et al., 2026). None of these methods states a finite-sample bound on deployed quality. We measure how many blocks the engine keeps in flight and the three levers of the serving loop (training, allocation and commitment) on the engine and its decode traces, each against its control. The engine keeps only about two blocks in flight because its deeper blocks wait for text that is not yet generated. The default commit rule already commits every fully resolved block once the blocks before it are committed, and a static skip rule saves nearly all of the compute that allocation can save. Self-distillation on targets that the engine itself decodes gains 5 to 6% TPF at unchanged accuracy. Larger gains come from lower thresholds, which let the engine commit positions that are still uncertain and condition later blocks on them. A learned checkpoint thus becomes one more candidate for the serving grid, and its risk, like that of a hand-picked setting, shows only in the answers.

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.

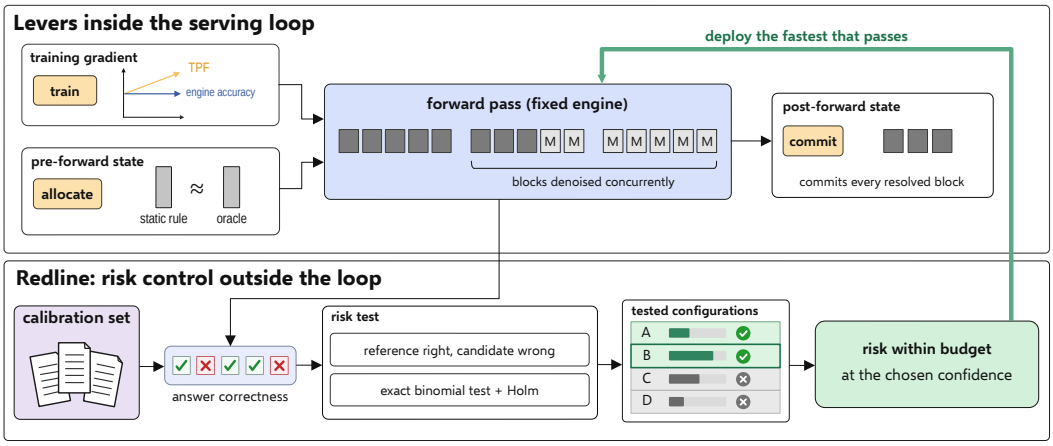


Figure 1: Levers inside the serving loop and risk control outside it. Top, the three levers at the decision sites they act on. Bottom, Redline tests every configuration on the correctness of its answers, keeps those that pass (green) and deploys the fastest.

Whether an answer is correct can be checked on held-out prompts after serving, and these outcomes suffice to control the risk of an operating point from outside the loop, as Figure 1 shows. For each candidate configuration, we define its reference-relative risk as the *joint* probability, over prompts, that a reference configuration answers correctly and the candidate does not. We present Redline, a procedure that tests every configuration of a searched grid against a user-chosen risk budget with an exact binomial test and a multiplicity correction, and deploys the fastest configuration that passes. The risk of the deployed configuration then stays within the budget at a user-chosen confidence level, in finite samples and without distributional assumptions. Because the risk counts only the prompts that the candidate newly fails, which a user switched to it meets as regressions, it also bounds the net accuracy drop from above. Redline needs no gradient update and no architectural change. Prior distribution-free risk control in decoding calibrates a single early-exit threshold (Angelopoulos et al., 2025; Schuster et al., 2022). We instead select from a searched grid over several serving settings, across which risk is not monotone.

We apply Redline to two BD-LM families on math and code. In both families, math gains speed at a smaller risk budget than code, and at every budget at which code gains speed, math does too. At a risk budget of 0.10, for instance, the deployed LLaDA2 math configuration commits over a third more tokens in each forward than the reference. Added to the SDAR math grid on MATH-500, the distilled checkpoint is deployed at every budget from 0.18 on, with a TPF gain of 46.7% against 39.5% for the best hand-picked configuration.

On the same prompts and at a budget of 0.10, a mean-accuracy rule, which deploys the fastest configuration whose accuracy is within a tolerance of the reference's, exceeds the risk budget in most data splits on SDAR math whenever its tolerance keeps pace with Redline on LLaDA2 math, and so does bounding the net accuracy drop instead. In contrast, Redline stays within its stated failure probability under both the held-out and the pooled measure of risk. Finally, because Redline uses only the correctness of answers and a cost measure, it applies unchanged to the acceptance rule of speculative decoding and to weight quantization, two lossy accelerations of autoregressive serving.

Our contributions are as follows.

- We introduce Redline, a distribution-free, finite-sample procedure that deploys the fastest configuration whose reference-relative risk it can keep within a user-chosen budget at a user-chosen confidence.

- We apply it to two block-diffusion families on math and code, show on the same data that mean accuracy hides the regressions that Redline bounds, and transfer it unchanged to speculative decoding and weight quantization.

- We measure the three in-loop serving levers, each against its control, and find that the large TPF gains depend on lower thresholds, while a checkpoint distilled in the loop adds 5 to 6% TPF at unchanged accuracy and is tested by Redline like any hand-picked configuration.

2 Background and Setup

2.1 Block-diffusion LMs and multi-block serving

Block-diffusion LMs factorize a sequence into fixed-size blocks that are generated autoregressively, while the tokens inside each block are denoised in parallel under a block-causal mask (Arriola et al., 2025), and the block boundary lets generated blocks be cached like an autoregressive prefix. The serving stack of Jin et al. (2026) extends this to a multi-block regime and raises TPF by up to $\sim$78% at equal or better accuracy. Several blocks are *in flight* at once, denoised concurrently inside a fixed-slot *block buffer*. The blocks in flight are governed by the admission threshold τ_{add}, the semi-completion threshold τ_{semi}, which lets later blocks condition on the top-1 context of a predecessor before it is cached, and the acceptance threshold τ_{acc}, together with the stability and revision thresholds of the engine. We keep the engine fixed throughout.

We measure parallelism by TPF, the number of tokens committed in each model forward. TPF is a *count ratio*, so it does not depend on host speed, kernel choice or batching policy. Its speculative-decoding analogue is the number of accepted tokens for each target forward. In the serving logs of our grids, TPF orders the configurations on each host exactly as measured end-to-end throughput does, with a Spearman correlation of 1.0, and Appendix F.4 reports these measurements.

2.2 Distribution-free risk control

A bounded prompt-level $0/1$ loss induces a *risk function* $R : \Lambda \to [0, 1]$, each configuration receives a *super-uniform* p*-value*, *family-wise error control* covers the m configurations tested at once, and the *Learn-Then-Test* framework (Angelopoulos et al., 2025) combines them into a set of configurations whose risks are controlled simultaneously. Our p-value is an exact binomial tail, with no asymptotics and no distributional model. We detail the formalism and the notation in Appendix A.

2.3 The serving-configuration space

A deployed block-diffusion engine has a multi-dimensional configuration, spanning accept and semi-completion thresholds, block-add and participation schedules, buffer depth, checkpoint and numeric precision. Published systems pick points in this space by hand. For example, LLaDA2.0 is released at one point with a reported $2.1\times$ speedup, LLaDA2.1 offers a menu of speed and quality modes, and SDAR reports a sweep over block sizes (Bie et al., 2025; 2026; Cheng et al., 2026).

Two observations about this space motivate the measurements that follow. First, on math decode traces of GSM8K and MATH-500, roughly 37% of active block-forwards in both families are no-ops that commit no token, and most of them fall on freshly added slots that are still fully masked. Second, only about two blocks stay in flight. A four-block buffer has nearly four still-masked slots in its forward window on an average step, yet the scheduler admits only about two blocks, and at most two are partially resolved in over 99% of steps in both families. Halving or doubling the buffer barely changes this saturation, which Jin et al. (2026) observe for naive multi-block decoding but leave unexplained. Appendix F gives the measurements for each family and the buffer sweep.

2.4 Three levers of the serving loop

An intervention inside the serving loop can act at three points, and each point defines one lever. The training gradient can *train* the model to commit more tokens in each forward (Chen et al., 2026a; Qian et al., 2026; Wang et al., 2026; Lee et al., 2026c), the pre-forward state can *allocate* forwards away from wasted slots (Lee et al., 2026a), and the post-forward state decides which resolved tokens to *commit* (Chen et al., 2026b; Bie et al., 2026). Engine redesign and cross-request scheduling change the loop itself and lie outside our scope.

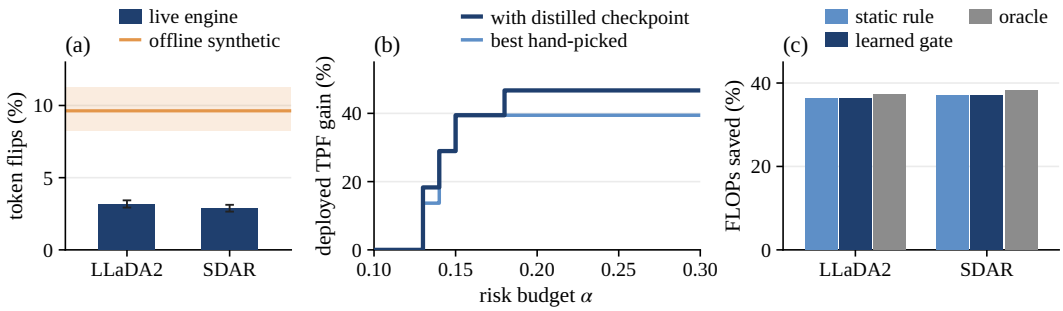


Figure 2: Measurements of the serving loop. (a) Token flips when unresolved predecessors are finalized. (b) Deployed TPF gain of Redline on MATH-500 with the distilled checkpoint, against the best hand-picked configuration. (c) FLOPs saved in the compute-bound regime. Whiskers give 95% intervals.

3 Measuring the Three Levers

Figure 2 summarizes our measurements of the blocks in flight and of the three levers on the engine and its decode traces, and Appendix B gives the protocol and the control of each.

3.1 Why only two blocks stay in flight

Only about two blocks stay in flight because the masked slots of deeper blocks remain uncertain. We re-run each committing forward inside the engine's own decode traces with its unresolved predecessor finalized to its eventual tokens. As Figure 2a shows, in both families this flips only about 3% of the positions committed while an older block still holds masks, roughly a third of the offline rate on synthetically masked states. In the same SDAR traces, the masked slots of the deeper blocks carry more predictive entropy than those of the first active block in 96.7% of the decode steps that hold a deeper block, and they still do in 67.3% once every predecessor is finalized, which removes about three fifths of their excess entropy. During serving, only 2.6 and 3.0% of the masked slots in the second and third blocks reach the accept threshold of 0.95, against 11.3% in the first.

3.2 The training lever: self-distillation on engine-decoded targets

We distill the model toward committing more tokens in each forward, on multi-block targets that the engine decodes with its own confidence rule. Table 1 compares the distilled checkpoint with the base checkpoint on held-out GSM8K prompts. At every admission threshold, it gains 5.2 to 5.8% TPF while accuracy changes by 0.0 to +0.8 points. We then serve it at the eight threshold settings of the SDAR math grid and test it beside the eight hand-picked configurations on the MATH-500 half of the math set, which neither the training nor the selection of the checkpoint touched. As Figure 2b shows, at every risk budget from 0.18 on, Redline deploys a configuration of the distilled checkpoint, with +46.7% TPF against +39.5% for the best hand-picked configuration on the same half.

Table 1: Self-distillation on engine-decoded targets, SDAR on 256 held-out GSM8K prompts. Engine accuracy and TPF at each admission threshold τ_{add}, with the change in points and in percent.

	engine accuracy			**engine TPF**		
τ_{add}	**0.05**	**0.10**	**0.20**	**0.05**	**0.10**	**0.20**
base	0.895	0.895	0.902	3.20	3.18	3.19
distilled	0.898	0.902	0.902	3.36	3.37	3.37
change	+0.39	+0.78	+0.00	+5.2	+5.8	+5.8

3.3 The allocation lever: a static skip rule

The no-op forwards of Section 2.3 invite a participation schedule that skips them. In a replay of the engine's decode traces, a static rule that skips a fully masked block while an older block is still active saves 36 to 37% of FLOPs in the compute-bound regime. As Figure 2c shows, this lies within 1.3 points of an oracle that removes every zero-commit forward, and a learned participation gate matches the static rule to within 0.02 points.

3.4 THE COMMITMENT LEVER: DEFAULT COMMITMENT IS MAXIMAL

Under the engine's default commit rule, every fully resolved block preceded only by committed blocks enters the cache in the same step, and no block that is safe to commit is left waiting. We prove this maximal-exact-commit invariant from the scheduler in Appendix C, and it holds at all 281,005 measured block-steps, whereas the revision-capable samplers of LLaDA2.1 and DMax (Bie et al., 2026; Chen et al., 2026b) can defer a completed block by one step.

4 RISK-CONTROLLED CONFIGURATION SELECTION

As Section 3 shows, the default commit rule already commits every fully resolved block, and the masked slots of deeper blocks seldom reach the accept threshold. A configuration that is faster at a fixed checkpoint therefore accepts tokens at lower confidence, and it can fail on prompts that the reference answers correctly. We build Redline from the correctness of each answer on held-out calibration prompts, from which the risk of a fixed operating point can be bounded with finite-sample confidence and without distributional assumptions, provided that the calibration prompts are drawn i.i.d. from the deployment distribution. Because each outcome records the realized correctness of a served output, the risk also covers the engine's run-to-run variation.

4.1 RISK AND COST

For each candidate configuration λ in the serving-configuration space, we define the reference-relative risk

$$R(\lambda) \;=\; \Pr_{x\sim\mathcal{D}}\big[\,\text{reference}(x)\ \text{correct}\ \wedge\ \lambda(x)\ \text{incorrect}\,\big], \tag{1}$$

the probability that the reference answers a prompt correctly and λ does not, which is the population form of the negative flip rate that model-update studies track (Yan et al., 2021; Xie et al., 2021). A user who is switched to λ meets each such failure as a regression, whatever λ fixes on other prompts (Bansal et al., 2019). The risk also upper-bounds the net accuracy drop $R - G$, where $G = \Pr(\text{reference incorrect} \wedge \lambda \text{ correct}) \geq 0$, and some faster configurations even gain net accuracy. Table 2 reports the net accuracy change beside each gain.

The reference, the conservative baseline of each space, is the engine's default configuration, lossless verification or bf16 weights, depending on the method. The cost, whether TPF, accepted tokens for each target forward or weight memory, serves only to choose among the configurations that pass, so the guarantee attaches to R alone, and we report the TPF *of* the deployed configuration, not a guaranteed TPF. Where TPF and verbosity could mix, we also report the number of forwards for each request.

4.2 REDLINE IN FOUR STEPS

Redline instantiates Learn-Then-Test (Angelopoulos et al., 2025) for serving, with the reference-relative joint risk of Eq. 1 in place of a marginal error rate. (i) *Fix* a grid Λ of m configurations that includes the reference, n i.i.d. calibration prompts, a risk budget α and a failure probability δ, with $\delta = 0.10$ throughout. (ii) *Score* each configuration by counting the prompts that the reference gets right and λ gets wrong, V_λ, and form the one-sided exact binomial p-value $p_\lambda = \Pr[\mathrm{Bin}(n, \alpha) \leq V_\lambda]$ for $H_0 : R(\lambda) > \alpha$. (iii) *Test* with the Holm step-down procedure (Holm, 1979) at family-wise level δ, which sorts $p_{(1)} \leq \cdots \leq p_{(m)}$ and rejects while $p_{(i)} \leq \delta/(m - i + 1)$, stopping at the first failure. We call the configurations whose null hypothesis is rejected *valid*. (iv) *Deploy* the fastest valid configuration $\lambda^\star$, or for quantization the one with the least memory. Because Holm controls the family-wise error over all valid configurations at once, the deployed configuration satisfies

$$\Pr\big(R(\lambda^\star) \leq \alpha\big) \;\geq\; 1 - \delta \tag{2}$$

in finite samples and without distributional assumptions, whichever valid configuration the cost selects, as Appendix A.5 shows.

We use a Holm-class correction because shared calibration prompts make the p-values dependent in an uncontrolled way, and Holm tolerates arbitrary dependence. Measured risk is also non-monotone across the grid, since the configurations with the strictest accept threshold run slower than the

Table 2: Configurations that Redline deploys at three risk budgets ($\delta = 0.10$, Holm), with the gain over the reference in the group's cost measure and the net accuracy change in points. Below $\alpha_{\min}$, the smallest budget at which a cheaper configuration passes, the reference is deployed.

				$\alpha = 0.10$		$\alpha = 0.15$		$\alpha = 0.20$	
grid	n	m	$\alpha_{\min}$	**gain**	**Δacc.**	**gain**	**Δacc.**	**gain**	**Δacc.**
Block-diffusion serving, TPF gain (%)									
LLaDA2 math	1,012	7	0.07	+37.5	−4.0	+37.5	−4.0	+37.5	−4.0
LLaDA2 math, large grid	1,012	50	0.07	+65.4	−3.4	+72.0	−5.9	+72.0	−5.9
LLaDA2 code	542	8	0.11	0.0	0.0	+33.7	−7.0	+42.4	−13.1
SDAR math	1,012	8	0.10	+13.1	−2.5	+36.6	−4.6	+36.6	−4.6
SDAR code	664	8	0.11	0.0	0.0	+25.7	−5.0	+59.7	−14.0
Speculative decoding on GSM8K, gain in accepted tokens for each target forward (%)									
Llama	512	7	0.09	+13.8	−3.3	+16.8	−9.0	+16.8	−9.0
Qwen	512	7	0.04	+18.1	−1.6	+18.1	−1.6	+18.1	−1.6
Weight quantization on GSM8K, reduction in weight memory									
Llama	512	4	0.07	2.81×	−1.4	2.81×	−1.4	2.81×	−1.4

reference yet still carry nonzero risk, so fixed-sequence testing, which walks a data-independent order, serves only as a secondary procedure. By construction, every configuration valid under Bonferroni is valid under Holm, and on our grids Holm deploys a faster configuration than Bonferroni at several budgets. Appendix D.3 compares the three corrections on every grid.

The guarantee holds at each budget with probability at least $1 - \delta$, and a similar bound covers the smallest budget at which a task gains speed, which we read off the grid of budgets. Gaining speed at a budget below the smallest true risk among the F configurations faster than the reference requires one of them to clear a Holm threshold of at most δ/F, which a union bound limits to probability δ.

The calibration size sets how close to the budget a configuration can sit and still pass reliably. With eight configurations, one that sits two points under a budget of $\alpha = 0.05$ passes with probability at least 0.8 from about $1{,}000$ prompts. Because nothing in Redline inspects diffusion internals, it needs only a reference configuration, a grid of lossy settings, correctness outcomes and a cost measure, so we also apply it unchanged to speculative-decoding acceptance and weight quantization.

5 Experiments

5.1 Setup

Models and engines. We apply Redline to LLaDA2-mini (Bie et al., 2025) and SDAR-8B (Cheng et al., 2026) on the multi-block engine (Jin et al., 2026), with the SDAR accept threshold made configurable, each over an accept × semi-completion grid served under a $4{,}096$-token generation budget with greedy decoding. The reference of both families is the engine's default, accept 0.95 with semi-completion 0.90, and Appendix F.3 lists every configuration.

Benchmarks and scoring. Math is the same fixed composite in both families, GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) with $n = 1{,}012$ prompts, and code pairs HumanEval+ with an MBPP variant, with $n = 542$ for LLaDA2 and 664 for SDAR (Chen et al., 2021; Austin et al., 2021; Liu et al., 2023). We score code by execution-based pass@1 and math by exact match.

Autoregressive methods. For speculative decoding, we test Llama-3.1-8B with a Llama-3.2-1B draft and Qwen2.5-7B with a Qwen2.5-1.5B draft on GSM8K ($n = 512$) under the typical acceptance of Medusa (Cai et al., 2024), against lossless verification, whose risk is zero by construction. For quantization, we test bitsandbytes int8, nf4 and fp4 weights (Dettmers et al., 2022; 2023) against bf16.

5.2 Where each task gains speed

Table 2 shows the deployed configurations at three budgets, and Figure 3a traces the deployed gain over the whole range of budgets. In both families, math gains speed at a smaller budget than code,

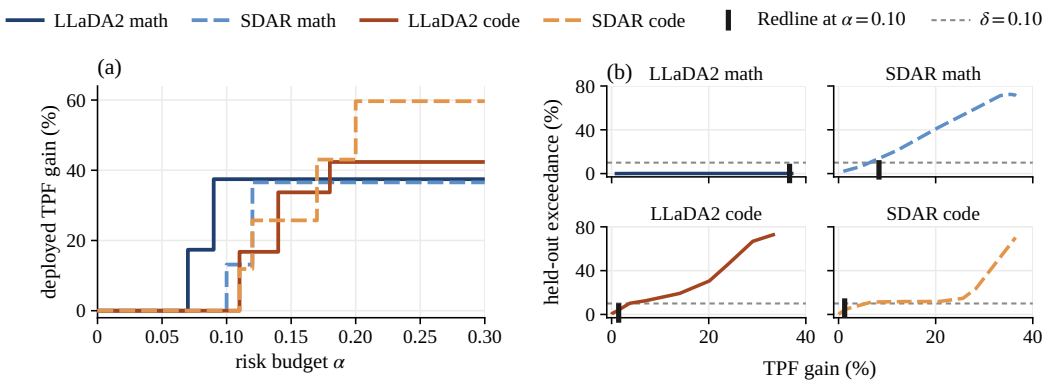


Figure 3: Redline and the mean-accuracy rule. (a) Deployed TPF gain across risk budgets. (b) Held-out exceedance, the rate at which the deployed configuration exceeds the budget on held-out halves, against TPF gain at $\alpha = 0.10$, for the mean-accuracy rule across its tolerances and for Redline.

and at every tested budget at which code gains speed, math does too. Every nonzero gain in TPF or accepted tokens carries a 95% paired-bootstrap interval that excludes zero. Both math configurations deployed at $\alpha = 0.10$ stay the same when every grid is tested again at $\delta = 0.05$ or 0.01, and hence also under a Bonferroni split of δ over the four grids.

At $\alpha = 0.10$, the deployed math configurations already run well ahead of the reference, at +37.5% TPF on LLaDA2 and +13.1% on SDAR, with 95% intervals of 34.0 to 41.1% and 7.1 to 19.6%. Both sit about three points of empirical risk under the budget, and the SDAR math configuration also cuts the forwards of each request by 15.9%.

Code first gains speed at $\alpha = 0.11$ in both families, and the ordering holds configuration by configuration. Each of the five configurations faster than the reference fails on 1.7 to 2.9 times as large a share of the reference's correct answers on code as on math. With math subsampled to the size of the code set, the smallest math budget with a gain stays below the code one in every LLaDA2 draw and at or below it in 99.1% of SDAR draws. Appendices D.1, D.3 and D.5 give the intervals, both comparisons and the forwards.

5.3 Against the mean-accuracy rule

Figure 3b compares Redline with the rule it replaces, selection by mean accuracy, on the same grids and prompts. A mean lets the prompts that a configuration fixes offset those it fails, and the SDAR math configuration deployed at $\alpha = 0.10$, for instance, turns a correct reference answer into a wrong one on 7.1% of prompts, nearly three times its net accuracy drop. Over 1,000 random half splits of each calibration set, we apply each rule to one half and score its deployed configuration on the other. We call the rate at which the risk R of that configuration exceeds the budget, on the test half or pooled over all prompts, the exceedance.

The mean-accuracy rule deploys the fastest configuration whose calibration-half accuracy is within t points of the reference's, for $t \in \{0, 1, \ldots, 8, 10\}$. At $\alpha = 0.10$, no single tolerance is both as fast as Redline on LLaDA2 math and as rarely over the budget on SDAR math. Every tolerance up to four points gains less TPF than Redline on LLaDA2 math, and every tolerance from five points up exceeds the budget in over 70% of held-out splits on SDAR math, against 3.4% for Redline. The trade-off persists when each rule is applied to 30 or 70% of the prompts instead of half. A tolerance chosen separately for each grid could be checked against the budget only by measuring R on that grid, which a mean does not report.

Each ingredient of Redline is needed. Dropping the multiplicity correction raises the held-out exceedance on SDAR math at $\alpha = 0.10$ from 3.4 to 17.5%, and dropping the finite-sample margin as well, the plug-in rule $\hat{R} \leq \alpha$, exceeds the budget in up to 64.1% of held-out splits. Testing the net drop instead of the joint risk, with the same Holm step and a betting p-value (Waudby-Smith & Ramdas, 2024), keeps that drop within budget in every split on SDAR math at $\alpha = 0.10$, yet its

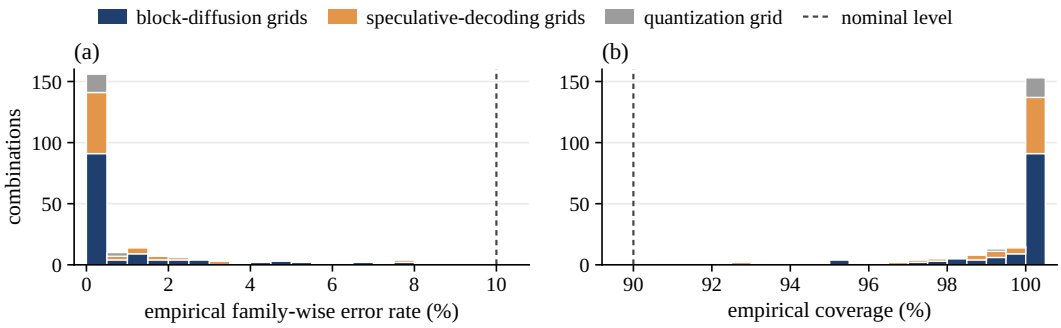


Figure 4: Out-of-sample validity of Redline over 220 combinations of grid, risk budget and calibration size, each on 1,000 random splits. (a) Empirical family-wise error rate. (b) Empirical coverage.

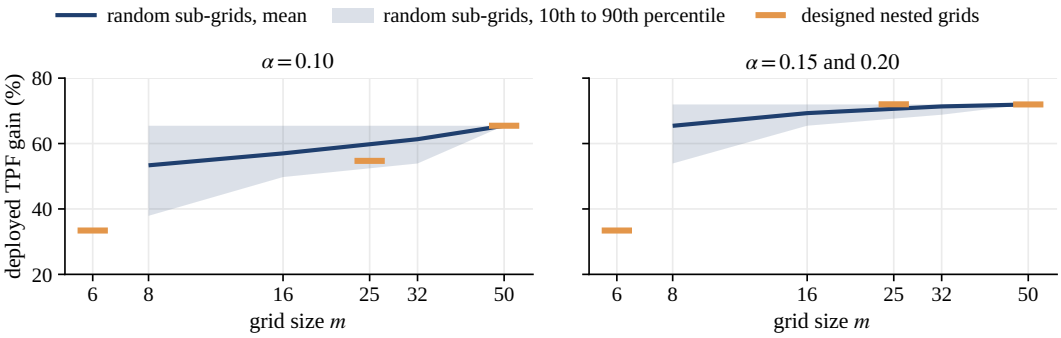


Figure 5: Deployed TPF gain of Redline against grid size on the LLaDA2 math grid of 50 configurations ($n = 1{,}012$, Holm, $\delta = 0.10$), for 1,000 random sub-grids of each size that hold the reference and for the designed nested grids.

deployed configurations exceed the joint budget in 91.4% of splits against the pooled risk. Redline itself stays under 7% held-out exceedance and at or under 0.2% pooled for all sixteen pairs of grid and budget. Appendix D.9 lists every pair and every rule.

5.4 Speculative decoding and quantization

Table 2 also lists the deployed configurations of two public autoregressive methods. For the typical acceptance of Medusa, a faster setting is deployed from $\alpha = 0.04$ on for the Qwen pair, with a net accuracy gain at $\alpha = 0.05$, and from $\alpha = 0.09$ on for the Llama pair. The Llama draft agrees with the target's argmax on more tokens than the Qwen draft, 90.5% against 88.7%, yet needs a larger budget, since the risk measures whether a divergent token changes the final answer rather than how often tokens diverge. Under quantization, all three low-precision formats are valid at $\alpha = 0.10$, and the deployed format, nf4, reduces weight memory by a measured factor of 2.81 and peak memory during generation by 2.75. Appendix E details both methods.

5.5 Validity and larger grids

As Figure 4 shows, across 12 grids and 220 combinations, repeated random calibration and test splits give a minimum coverage of 0.904 and a maximum family-wise error rate of 0.096, and no combination exceeds the nominal level. When we run Redline on one random half of each calibration set, the deployed configurations re-measured on the other half stay close to their in-sample gains, for instance +37.6% against +37.5% TPF for the LLaDA2 math configuration at $\alpha = 0.10$ and +13.4% against +13.1% for the SDAR math configuration, with net accuracy changes within a point of the in-sample values. Appendices A.7 and D.10 give the split protocol.

A joint grid over the accept, semi-completion and add thresholds deploys exactly the LLaDA2 math configuration of Table 2, and on Fast-dLLM-v2 a faster accept threshold is deployed at $+15.3\%$ TPF at $\alpha = 0.10$. On a separate LLaDA2 math grid of 50 configurations over all three lossy thresholds of the engine, listed in Table 2 with its own reference, 36 of the 50 configurations are valid at $\alpha = 0.10$ and all 50 at $\alpha \geq 0.15$. Its deployed configurations reach $+65.4\%$ and $+72.0\%$ TPF, with a held-out exceedance of 5.3% at $\alpha = 0.10$ and none at 0.15 and 0.20, where every split deploys the same configuration. As Figure 5 shows, random sub-grids deploy faster configurations as they grow, despite the stricter correction. Appendices D.7, D.8 and D.11 give these three grids.

6 Related Work

Block-diffusion LMs. Masked-diffusion LMs (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025) established the discrete-diffusion alternative to autoregressive decoding, and BD3-LM (Arriola et al., 2025) is the architectural ancestor of our setting. SDAR (Cheng et al., 2026) and LLaDA2.0 and LLaDA2.1 (Bie et al., 2025; 2026) are the families we evaluate, and their published operating points come without a prompt-level guarantee. We serve on the multi-block engine of Jin et al. (2026).

Efficient and parallel decoding for diffusion LMs. Fast-dLLM and Fast-dLLM-v2 (Wu et al., 2026b;a), D2F (Wang et al., 2026), AdaBlock-dLLM (Lu et al., 2026), dParallel (Chen et al., 2026a), LoPA (Xu et al., 2025), d3LLM (Qian et al., 2026), LightningRL (Hu et al., 2026) and DMax (Chen et al., 2026b) push the same frontier by training, scheduling or caching. None of them states a distribution-free finite-sample bound on deployed quality or controls multiplicity over a configuration space. The nearest statement is Theorem 1 of Fast-dLLM, which is deterministic, conditional on confidence and has no δ. The thresholds and checkpoints of these methods are *inputs* to our space rather than alternatives to it.

Distribution-free risk control. Learn-Then-Test (Angelopoulos et al., 2025) is our testing framework, and conformal risk control (Angelopoulos et al., 2024) bounds the expected loss rather than the probability of exceeding the budget. CALM (Schuster et al., 2022), the closest prior use in decoding, calibrates one early-exit threshold of autoregressive decoding against the full model by a fixed-sequence walk. Redline selects from a searched grid of serving configurations whose risk is non-monotone in the thresholds, which Holm handles without an order, and applies unchanged to any cost measure. To our knowledge, no concurrent work selects serving configurations by distribution-free multiple testing, and the nearest analogs are Farzaneh & Simeone (2026) and Cai & Li (2026).

Regressions under model updates. Model-update studies train a new model to lower its negative flip rate against an old one (Yan et al., 2021; Xie et al., 2021). We instead bound that rate below a budget for the serving configurations of a fixed model, without training.

Lossy autoregressive acceleration. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) and the typical acceptance of Medusa (Cai et al., 2024) supply the acceptance setting and quantization the memory setting, both of which Redline calibrates. Appendix G compares these methods with our guarantee.

7 Discussion & Conclusion

Limitations. The guarantee is relative to the calibration distribution and to a reference configuration, so a deployment that serves a different prompt distribution or batching policy re-calibrates first. Our experiments cover greedy decoding on math and code in two block-diffusion families, and Redline applies to any task with verifiable correctness. Each guarantee holds at its own level $1 - \delta$, and several guarantees hold jointly when δ is split across them.

Conclusion. Redline makes the operating points of an engine, hand-picked or learned, selectable at a stated risk. A checkpoint trained with any signal, such as self-distillation or verifier-rewarded reinforcement learning, joins the grid as one more set of configurations, and Redline itself needs no training and carries over unchanged to any lossy serving method with a cost measure. On four grids it orders math before code in the same way across two families, and it bounds the regressions that mean accuracy hides behind fixes.

Reproducibility Statement

Every guarantee and deployment in this paper is a deterministic function of prompt-level correctness counts and configuration-level cost counts. Appendix D prints these counts for every configuration of the six block-diffusion grids and Table 4 for the grid with the distilled checkpoint, so their valid sets and deployments can be re-derived without the serving engine. The supplementary material contains the code of Redline and the prompt-level outcomes from which the deployments of Table 2 and the validity check of Figure 4 can be recomputed. Appendix A states Redline formally with the assumptions it needs, Appendix F gives the full engine configuration together with the one engine change that the SDAR grids need, and Appendix B gives the protocol of every measurement. All grid runs decode greedily, so no sampling seed is involved.

AI Use Statement

A large language model assisted with LaTeX tables, and language polishing. The authors reviewed and verified all outputs and take full responsibility for the paper.

Ethics Statement

We select serving configurations of existing open-weight language models and introduce no new model, dataset or study with human subjects. All benchmarks are public (GSM8K, MATH-500, HumanEval, HumanEval+, MBPP and MBPP+) and are used under their licenses. The guarantee holds for the calibrated prompt distribution, and a deployment that serves a different distribution re-calibrates before relying on it.

References

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f3549ef9b5ff520a7e41ff3cc306ab2b-Abstract-Conference.html.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025. doi: 10.1214/24-AOAS1998.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7ede97c3e082c6df10a8d6103a2eebd2-Abstract-Conference.html.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pp. 2429–2437, 2019. doi: 10.1609/aaai.v33i01.33012429.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6), 2021. doi: 10.1145/3478535.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, et al. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025. URL https://arxiv.org/abs/2512.15745.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, et al. LLaDA2.1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026. URL https://arxiv.org/abs/2602.08676.

Changxiao Cai and Gen Li. Confidence-based decoding is provably efficient for diffusion language models. *arXiv preprint arXiv:2603.22248*, 2026. URL https://arxiv.org/abs/2603.22248.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/57250222014c35949476f3f272c322d2-Abstract-Conference.html.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. DMax: Aggressive parallel decoding for dLLMs. *arXiv preprint arXiv:2604.08302*, 2026b. URL https://arxiv.org/abs/2604.08302.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026. doi: 10.18653/v1/2026.findings-acl.1110.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2208.07339.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.14314.

Amirmohammad Farzaneh and Osvaldo Simeone. Statistically valid post-training hyperparameter selection: From tuning to guarantees. *arXiv preprint arXiv:2606.25601*, 2026. URL https://arxiv.org/abs/2606.25601.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Yanzhe Hu, Yijie Jin, Pengfei Liu, Kai Yu, and Zhijie Deng. LightningRL: Breaking the accuracy–parallelism trade-off of block-wise dLLMs via reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/65221.

Yijie Jin, Jiajun Xu, Yuxuan Liu, Chenkai Xu, Yi Tu, Jiajun Li, Dandan Tu, Xiaohui Yan, Kai Yu, Pengfei Liu, and Zhijie Deng. Multi-block diffusion language models. *arXiv preprint arXiv:2606.29215*, 2026. URL https://arxiv.org/abs/2606.29215.

Jungseob Lee, Seongtae Hong, Seungjun Lee, Jaehyung Seo, Junyoung Son, Sugyeong Eo, Chanjun Park, Hyeongju Park, Hyeonseok Moon, and Heuiseok Lim. DART: Draft-agreement routing for training-free adaptive thinking budgets in hybrid reasoning models. *arXiv preprint arXiv:2606.23181*, 2026a. URL https://arxiv.org/abs/2606.23181.

Jungseob Lee, Seungyoon Lee, Seongtae Hong, Minhyuk Kim, Chanjun Park, and Heuiseok Lim. Beyond penalizing mistakes: Stabilizing efficiency training in large reasoning models via adaptive correct-only rewards. *arXiv preprint arXiv:2606.22716*, 2026b. URL https://arxiv.org/abs/2606.22716.

Jungseob Lee, Seungyoon Lee, Suhyune Son, Dongyub Jude Lee, Sungbin Han, Sugyeong Eo, and Heuiseok Lim. Answer-conditioned chains of thought degrade verifiable-reasoning distillation in large language models. *arXiv preprint arXiv:2607.14552*, 2026c. URL https://arxiv.org/abs/2607.14552.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

Guanxi Lu, Hao Mark Chen, Yuto Karashima, Zhican Wang, Daichi Fujiki, and Hongxiang Fan. AdaBlock-dLLM: Semantic-aware diffusion LLM inference via adaptive block size. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e2eeb89acc98e8e506d719e330cbc43a-Abstract-Conference.html.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3LLM: Ultra-fast diffusion LLM using pseudo-trajectory distillation. In *International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/61269.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2207.07061.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/04bb76a98d9f32226b3beba7bd26a51f-Abstract-Conference.html.

Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(1):1–27, 2024. doi: 10.1093/jrsssb/qkad009.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d0865cbe51d35ec322f9af9db7806fc7-Abstract-Conference.html.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d8d4e6061c3ba96c240b7fa1ae3471d-Abstract-Conference.html.

Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. Regression bugs are in your model! Measuring, reducing and analyzing regressions in NLP model updates. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 6589–6602, 2021. doi: 10.18653/v1/2021.acl-long.515.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, and Zhijie Deng. LoPA: Scaling dLLM inference via lookahead parallel decoding. *arXiv preprint arXiv:2512.16229*, 2025. URL https://arxiv.org/abs/2512.16229.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14294–14303, 2021. doi: 10.1109/CVPR46437.2021.01407.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. URL https://arxiv.org/abs/2508.15487.

Table 3: Notation used throughout the paper.

λ, Λ	a serving configuration and the finite grid of configurations searched, spanning accept and semi-completion thresholds, block-add schedule, buffer depth, checkpoint and precision
reference	the conservative baseline configuration of each grid, which is the engine's default for block diffusion, lossless verification for speculative decoding and bf16 weights for quantization
$R(\lambda)$	reference-relative risk, the *joint* probability that the reference answers a prompt correctly and λ does not. It upper-bounds the net accuracy drop
α	the risk budget. The guarantee asserts $R(\lambda) \leq \alpha$
δ	the failure probability. All statements about one grid hold jointly with probability at least $1-\delta$, and $\delta = 0.10$ throughout
TPF	tokens committed in each model forward, a count ratio independent of host speed. The speculative-decoding analogue is the number of accepted tokens for each target forward
n	the number of calibration prompts of a grid
acc85/semi70	a configuration label, here accept threshold 0.85 and semi-completion threshold 0.70. A suffix such as add0.30 gives the admission threshold

A Risk-Control Formalism and Validity

A.1 Setup and the guarantee

Let $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$ be the finite grid of serving configurations, evaluated on n i.i.d. calibration prompts. Each configuration carries the bounded prompt-level loss of Section 4.1, the $0/1$ reference-relative joint loss with $L_i(\lambda) = 1$ exactly when the reference decode is correct on prompt i and the decode of λ is not, and the risk is $R(\lambda) = \mathbb{E}[L(\lambda)]$. The loss is the realized correctness of the served output, so the expectation runs over the prompt and over the serving randomness of the engine under the serving policy of the calibration runs, batch composition included. The i.i.d. assumption is on prompts drawn together with their realized outcomes. The guarantee therefore covers the configuration as it is served under that policy, and a different batching policy is a distribution shift that is re-calibrated like any other. Redline follows Learn-Then-Test and returns a valid set inside $\{\lambda : R(\lambda) \leq \alpha\}$ with family-wise error control,

$$\Pr\big(\exists\, \lambda \in \text{valid set} : R(\lambda) > \alpha\big) \;\leq\; \delta. \tag{3}$$

Two properties matter downstream. The guarantee holds for any loss distribution and any selection statistic, so the quality of the grid decides only which configurations pass. The statement is also simultaneous over the valid set, which is what makes the deployment rule of Appendix A.5 free.

A.2 The exact binomial p-value

For $H_0 : R(\lambda) > \alpha$ with Bernoulli losses, the violation count $S = \sum_i L_i(\lambda)$ satisfies $S \sim \mathrm{Bin}(n, R)$. The p-value

$$p(\lambda) = \Pr_{\mathrm{Bin}(n,\alpha)}\big(S \leq V(\lambda)\big), \qquad V(\lambda) = \text{observed violations}, \tag{4}$$

is super-uniform under H_0 by stochastic ordering. $\mathrm{Bin}(n, R)$ with $R > \alpha$ stochastically dominates $\mathrm{Bin}(n, \alpha)$, so small counts are rarer under every null than at the boundary $R = \alpha$. Rejecting at level δ therefore establishes $R \leq \alpha$ with error probability at most δ.

A.3 Holm step-down under arbitrary dependence

Order $p_{(1)} \leq \cdots \leq p_{(m)}$ and reject while $p_{(i)} \leq \delta/(m-i+1)$, stopping at the first failure. Holm (1979) shows that this controls the family-wise error rate at δ under **arbitrary** joint dependence of the p-values. Our setting needs exactly this property, because every configuration is scored on the *same* calibration prompts, which induces strong positive dependence. Holm rejects a superset of what Bonferroni rejects, since each ordered threshold $\delta/(m-i+1)$ is at least δ/m, and we keep Bonferroni as the conservative floor. Appendix D.3 quantifies the gap between the two.

A.4 FIXED SEQUENCE AS SECONDARY

The fixed-sequence procedure walks the configurations in the rank order of the design thresholds and tests each at full δ until the first failure. Its validity needs no monotonicity of the risk, because an ordering independent of the calibration losses suffices, and monotonicity affects only its power. We report it as secondary and never mix selections across procedures at one α. It is not primary because measured risk is **non-monotone** in the design thresholds. The configurations with accept threshold 0.99 in Appendix D.2 are slower *and* riskier than the reference, so any such walk can stop early on an out-of-order configuration, whereas Holm needs no order.

A.5 SIMULTANEITY OF THE DEPLOYMENT RULE

The deployed configuration is $\lambda^\star = \arg\max_{\lambda \text{ valid}}$ measured TPF. TPF is a point estimate that never enters any p-value, and the family-wise statement of Appendix A.1 covers every valid configuration **simultaneously**. The post-hoc choice of the fastest configuration therefore costs nothing,

$$\Pr\big(R(\lambda^\star) > \alpha\big) \;\le\; \Pr\big(\exists\, \lambda \in \text{valid set} : R(\lambda) > \alpha\big) \;\le\; \delta. \tag{5}$$

A.6 ERROR LEVELS WITHIN AND ACROSS GRIDS

The guarantee of each grid holds at confidence $1 - \delta$ with $\delta = 0.10$, and the paper presents its four primary block-diffusion grids individually and never pooled. When the conjunction is wanted, Bonferroni over the four grids governs, each grid running at $\delta/4 = 0.025$. Under that correction the SDAR math configuration deployed at $\alpha = 0.10$ is still valid, with $p = 8.5 \times 10^{-4}$ under the Holm threshold $0.025/6 \approx 0.0042$ at its rank.

The smallest budget with a gain. Section 5.2 reads, for each grid, the smallest budget $\hat{\alpha}$ on the budget grid at which a configuration faster than the reference is valid under Holm. No guarantee that holds jointly along the α grid is claimed, and none is needed for that one statement.

Let F be the number of faster configurations, five in each grid, and α_0 the smallest grid budget at or above the smallest true risk among them. If $\hat{\alpha} < \alpha_0$, then every faster configuration is a true null at $\hat{\alpha}$. Holm rejects a prefix of the sorted p-values, so for any faster configuration to be valid the first faster configuration in that order must clear its threshold, and since at most the reference and the slower configurations precede it, that threshold is at most δ/F. Each p-value decreases in α, so the event is contained in $\{\min_{\text{faster}} p_\lambda(\alpha_0^-) \le \delta/F\}$ at the single grid budget α_0^- just below α_0, where all F of those p-values are super-uniform, and the union bound gives probability at most δ. Hence $\Pr(\hat{\alpha} \ge \alpha_0) \ge 1 - \delta$ for each grid and $1 - 2\delta$ for the two math grids jointly. The statement concerns the budget and not the configuration valid at $\hat{\alpha}$, so deployments are reported at the three budgets of Table 2, and deploying the reference is a non-rejection that carries no statement about its task.

A.7 EMPIRICAL VALIDITY AT THREE LAYERS

The guarantee is checked at three layers, each catching what the previous one cannot. **Theorem.** Super-uniform p-values combined with Holm control the family-wise error under arbitrary dependence, as Appendices A.2 and A.3 show.

Synthetic Monte Carlo with known risk. Five configurations carry true risks $\{0.01, 0.02, 0.07, 0.07, 0.07\}$ at $\alpha = 0.05$, two below the budget and three above it, with $n = 800$ and 4,000 trials for each procedure. For Holm, the losses of all five configurations are drawn from a *shared* prompt-level uniform, which induces the positive dependence across configurations of the real setting, and the empirical family-wise error rate is 0.0003. Fixed sequence and Bonferroni, run on independent draws, read 0.

Out-of-sample coverage on real data. Twelve grids, $K = 1{,}000$ random calibration and test splits each, budgets $\alpha \in \{0.05, 0.10, 0.15, 0.20\}$ and two to five calibration sizes give 220 combinations, summarized in Figure 4. The twelve grids are the six block-diffusion grids of Appendix D, a LLaDA2 math grid on 32 prompts of each subtask, the speculative-decoding grids of both draft pairs on GSM8K and on HumanEval, and the quantization grid. The minimum coverage is 0.904, the maximum empirical family-wise error rate is 0.096, and no combination exceeds the nominal level. Measured against each valid configuration's risk pooled over all n prompts rather than its

test-half estimate, a proxy that the calibration half also informs, the same 220 combinations give a maximum family-wise error rate of 0.014, and no combination exceeds δ under either measure. This layer resamples a finite benchmark under exchangeability, so it complements the theorem rather than replacing it.

B MEASUREMENT PROTOCOLS

Each measurement of Section 3 is read against a control. Live predecessor finalization is compared with the engine's own traces during serving, self-distillation with the base checkpoint, and the static skip rule with a learned participation gate and an unconstrained oracle over the same features. Appendix C proves the commit invariant and checks it on the engine's traces.

B.1 LIVE PREDECESSOR FINALIZATION

Protocol. The measured quantity is the live flip rate. We call a committed position exposed when, at its step, the committing block has at least one older active block with at least one masked position. The flip rate is the probability over exposed positions that the argmax under the predecessor-finalized counterfactual differs from the token actually committed. A replay counts only when its recomputed argmax reproduces the committed token on at least 99% of the exposed positions, and every main replay passes this check at 99.39 to 99.70%. The main replay covers the requests that finish within the generation cap. Every live rate carries a Wilson 95% interval and a request-level cluster bootstrap 95% interval over $2{,}000$ resamples, since positions within a request correlate.

Live and offline readings. On the engine's own traces the flip rate is 2.88% of $117{,}731$ exposed committed positions on SDAR, with cluster interval $[2.65, 3.12]$, and 3.17% of $137{,}540$ on LLaDA2, with cluster interval $[2.93, 3.43]$. Adding the early decode steps of the 50 SDAR requests that reached the generation cap gives 2.90%. An offline synthetic factorial reads 9.63% $[8.24, 11.23]$ under the block-causal mask and 10.91% $[9.4, 12.6]$ under a bidirectional mask at $n = 1{,}500$, so the live rate is about a third of every synthetic reading in both families, and no live interval reaches a synthetic one. The synthetic settings force a joint masking corner that serving rarely visits. Live, 76 to 77% of exposed commits face a predecessor at most 25% masked, and that corner flips under 4%. Over all committed tokens, the identity-flip rate weighted by serving is 0.888% on SDAR and 0.876% on LLaDA2.

Entropy of the deeper blocks. The companion measurement asks what the masked slots of the deeper blocks lack while they wait. On the first 100 GSM8K requests of the same SDAR reference-configuration trace, 99 of which pass the reconstruction check, every decode step with at least two active blocks holding masked generation slots, $6{,}593$ steps in all, was forwarded again unchanged and, for each deeper block, with the masked slots of every older active block finalized to their eventual tokens. The predictive entropy of every masked slot was read from the mask-suppressed softmax that the engine commits from, and the argmax of the unchanged forward reproduces the committed token at 99.64% of exposed positions.

The statistic is the share of steps in which the slot-weighted mean entropy of the deeper blocks exceeds that of the first active block. It is 96.7% in the served state, with 95% request-cluster bootstrap interval $[95.8, 97.6]$ and 3.02 against 1.53 nats on average over steps, and 67.3% with predecessors finalized, with interval $[65.3, 69.3]$ and 2.10 nats. Finalizing the predecessors removes about three fifths of the excess entropy of the deeper blocks. By block rank, the share of masked slots at or above the accept threshold of 0.95 of the reference configuration is 11.3% in the first active block, 2.6% in the second and 3.0% in the third in the served state, and 15.2% and 21.4% in the second and third once their predecessors are finalized.

The committing forward responds to predecessor state at the masked slots, whereas the positions it commits keep their identity under finalization in 97% of exposed cases. About three fifths of the excess uncertainty of a deeper block is predecessor text that the engine supplies by generating it.

B.2 SELF-DISTILLATION ON ENGINE-DECODED TARGETS

Protocol. The training targets follow the engine's own decoding. They come from $1{,}000$ GSM8K prompts disjoint from the $n = 256$ evaluation prompts, with at most 12 blocks of 32 tokens, decoded

Table 4: Redline on MATH-500 ($n = 500$) over the eight hand-picked SDAR math configurations and the distilled checkpoint at the same threshold settings ($m = 16$). TPF, forwards and tokens of each request, joint risk $\hat{R}$ and p-values. Bold marks configurations valid under Holm, and underlining the deployed one.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
Hand-picked configurations of the SDAR math grid								
acc85/semi70	5.829	147.2	858	0.106 (53/500)	1.000	0.704	**0.003**	**1e-8**
acc90/semi70	5.490	164.5	903	0.118 (59/500)	1.000	0.919	**0.023**	**8e-7**
acc85/semi90	5.390	161.2	869	0.102 (51/500)	1.000	0.596	**0.001**	**3e-9**
acc90/semi90	4.752	177.9	845	0.094 (47/500)	1.000	0.361	**1e-4**	**9e-11**
acc95/semi70	4.724	195.7	924	0.114 (57/500)	1.000	0.867	**0.012**	**2e-7**
acc95/semi90 (reference)	4.180	210.5	880	0.000 (0/500)	**7e-12**	**1e-23**	**5e-36**	**4e-49**
acc99/semi70	3.273	256.0	838	0.100 (50/500)	1.000	0.538	**7e-4**	**1e-9**
acc99/semi90	3.189	259.6	828	0.060 (30/500)	0.869	**0.001**	**3e-10**	**6e-19**
Distilled checkpoint at the same eight threshold settings								
acc85/semi70	6.133	153.9	944	0.146 (73/500)	1.000	1.000	0.431	**0.001**
acc85/semi90	5.463	160.2	875	0.148 (74/500)	1.000	1.000	0.481	**0.002**
acc90/semi70	5.381	162.0	871	0.122 (61/500)	1.000	0.954	0.043	**3e-6**
acc90/semi90	4.945	173.8	860	0.092 (46/500)	1.000	0.306	**8e-5**	**4e-11**
acc95/semi70	4.826	181.6	876	0.110 (55/500)	1.000	0.796	**0.006**	**5e-8**
acc95/semi90	4.153	202.0	839	0.070 (35/500)	0.980	0.012	**3e-8**	**4e-16**
acc99/semi70	3.649	246.6	900	0.090 (45/500)	1.000	0.255	**4e-5**	**1e-11**
acc99/semi90	3.250	256.9	835	0.056 (28/500)	0.768	**3e-4**	**3e-11**	**4e-20**

iteratively within each block with the engine's confidence rule and with blocks strictly sequential. The rule decodes greedily at threshold 0.95 and falls back to the top-1 token when no token clears it. These completions reach 0.88 to 0.89 accuracy. Training runs for 400 single-block steps drawn from all $7{,}130$ blocks of these completions at learning rate 2×10^{-5}, and engine accuracy and TPF are read against the base checkpoint.

Result. Table 1 in Section 3.2 lists both checkpoints at the three admission thresholds. The distilled checkpoint gains $+5.2$, $+5.8$ and $+5.8\%$ TPF with accuracy changes of $+0.4$, $+0.8$ and $+0.0$ points, none of them significant, with $p \geq 0.84$.

Learn, then test. The distilled checkpoint was served at the eight threshold settings of the SDAR math grid on the same engine build and tested beside the eight hand-picked configurations as one grid of $m = 16$ hypotheses. Its training pool, GSM8K rows 256 to $1{,}255$, overlaps the GSM8K prompts of the math set, so the test uses the MATH-500 half ($n = 500$), which neither training nor selection touched, with speed and risk measured against the reference of the grid on the same prompts. Table 4 lists every configuration. The loosest threshold setting of the checkpoint, acc85/semi70, runs faster than every hand-picked configuration, and at every budget $\alpha \geq 0.18$, Redline, run on all sixteen configurations, deploys it at $+46.7\%$ TPF, with $\hat{R} = 0.146$ (73 of 500), against $+39.5\%$ for the best hand-picked configuration.

B.3 Participation schedules

Cost model. In both families, a participation schedule pays off when it cuts the FLOPs of each token without paying the saving back in extra steps. Wall-clock cost is modeled in token-equivalent units as steps $\cdot\, \rho +$ FLOP, where ρ is the fixed overhead of each forward divided by the compute of each token. At $\rho \to 0$, the compute-bound regime that a throughput claim targets, the speedup reduces to the FLOP saving, and at $\rho \to \infty$ to the ratio of step counts. The analysis sweeps the whole range without a GPU.

Replay. The schedules are replayed on the engine's decode traces with the committed tokens of every block held fixed, so that deferring the forward of a block changes only the step count. The static skip rule skips a fully masked block while an older block is still active. The learned participation gate scores the same causal pre-forward features, and the unconstrained oracle removes every zero-commit

forward. At $\rho \to 0$ the static rule saves 36 to 37% of FLOPs in both families, the oracle lies at most 1.3 points above it, and the learned gate matches the static rule to within 0.02 points, as Figure 2c shows.

C THE MAXIMAL-EXACT-COMMIT INVARIANT

C.1 STATEMENT

Proposition (maximal exact commit under default commit semantics). *Under the engine's default commit rule, the commit sweep of every decode step transfers to the cache, within that step, every active block that is fully resolved, commit-ready and preceded only by committed blocks. In the serving configurations deployed for both model families, a fully resolved block is always commit-ready when the sweep reaches it, so the exact-committable but uncommitted block mass is zero by construction.*

A block-step is *exact-committable but uncommitted* when a block is active, has no masked position and has no uncommitted predecessor, that is, a block the engine could commit at no risk yet does not. Under in-order commit semantics this is the right notion of committability. A complete block behind an uncommitted predecessor cannot be committed, because the cache must stay contiguous, and such steps are counted separately as legitimate in-order waits.

C.2 PROOF

The proof has three parts, each a property of the serving engine.

(1) The commit sweep is exhaustive within a step. The sweep visits the block buffer from left to right and commits a block exactly when it is active, complete, commit-ready and its predecessor is committed. Blocks are examined in strictly increasing order, and removing the first block shifts the indices so that no block is skipped or examined twice. The predecessor condition is monotone within a pass, because the transitions into and within the cache are one-way. Completeness and commit readiness cannot change during the sweep, since all sampler writes and state updates precede it. A freshly committed block satisfies its successor's predecessor condition in the same pass, and because each predecessor is examined first and its transitions are one-way, its status is final when its successor is examined.

Chains of ready blocks therefore commit in a single step, which matches the measured distribution of multi-block commits (295, 32 and 10 steps committing two, three and four blocks at once on an SDAR trace with a four-block buffer). Committed blocks form a prefix of the buffer under in-order commit, so removing the first block is benign. After every sweep, no block remains that is active, complete, commit-ready and preceded only by committed blocks.

(2) Complete implies commit-ready for both deployed samplers. The scheduler sets a block's commit readiness from a state map when the sampler provides one, and otherwise marks every complete active block commit-ready. The SDAR sampler provides no state map. The LLaDA2 samplers that do provide one are selected only by the in-place revision mode of LLaDA2.1 and by the token-merge decoding of DMax, and no deployed or measured configuration selects either. Every measured and tested configuration runs the plain LLaDA2 sampler, which provides no state map, so the executed path contains no deferral clause. The measured zero of Appendix C.3 also confirms the plain path, since the quiescence check of the revision sampler defers essentially every block completion by one step.

(3) Snapshots observe the state after the sweep. Each engine step runs the sweep before it returns, and the generation loop records the trajectory snapshot afterwards, with nothing changing the request state in between. A preempted request re-enters through steps that re-encode its prompt, which the check excludes, and the sweep runs in every step, so the invariant holds at every recorded decode snapshot.

Parts (1) to (3) together give the proposition. ■

C.3 Validation on decode traces

A check counts exact-committable but uncommitted block-steps over four decode traces, two families at buffer depths 2 and 4 on math with 128 prompts in each benchmark, and finds **none among** 281,005 **decode block-steps** (61,587, 66,811, 74,550 and 78,057). Its predicate ignores commit readiness altogether and counts blocks that are active, complete and preceded only by committed blocks, so the measured zero is stronger than the sweep invariant. It also rules out any commit-ready deferral and confirms the snapshot ordering of part (3).

C.4 Scope

The proposition covers default commit semantics. The revision-capable sampler modes, in-place revision in LLaDA2.1 and token merging in DMax, admit a one-step quiescence deferral and lie outside it, since in those modes the content of a block without masks can still change.

D Full Grids and Selector Comparisons

D.1 Grid tables

Tables 8 to 13 report, for every configuration of every grid, the measured TPF, the forwards and tokens of each request, the empirical joint risk $\hat{R} = k/n$ (reference correct and configuration wrong), the exact binomial p-value at each printed budget, whether the configuration is valid under Holm, and the deployed configuration.

Intervals for every deployed configuration. Table 5 attaches intervals to every deployed configuration other than the reference. The speed gain and the net accuracy change carry 95% paired-bootstrap percentile intervals over prompts with $B = 10{,}000$, the same resample weighting the deployed configuration and the reference, and the empirical joint risk carries its two-sided 95% Clopper–Pearson interval and the one-sided 90% upper bound that a single test at $1 - \delta$ would report. Across the 23 such entries at the four printed budgets, all 20 speed intervals lie above zero and every upper bound lies below its budget. The net intervals of the Qwen speculative-decoding deployments and of the quantized Llama include zero, so those deployments are not shown to cost accuracy at $n = 512$. The quantization gain is a measured property of the precision and carries no sampling error. The intervals are conditional on the configuration that the full-sample procedure selected, and Appendix D.10 re-measures the deployed configurations on held-out prompts.

D.2 Non-monotone risk

In all four main grids, risk is **non-monotone in the accept threshold**. The configurations with accept threshold 0.99 are *slower* than the reference yet carry nonzero reference-relative risk. On SDAR math, acc99/semi70 runs at TPF 3.09 against 3.85 for the reference with $\hat{R} = 0.084$ (85 of 1,012), and on LLaDA2 code, acc99/semi90 runs at 3.69 against 5.32 with $\hat{R} = 0.035$. A practitioner hand-tuning conservative thresholds therefore gets a configuration both slower and riskier than the default, with no signal saying so. Holm needs no order across the grid, which is why it is primary and why the fixed-sequence walk, which needs one, is secondary.

D.3 Deployed gain across budgets, and Holm against Bonferroni

Holm against Bonferroni. At identical (α, δ), Holm's valid set contains Bonferroni's by construction. Holm raises the deployed gain over Bonferroni's at grid-specific budgets between 0.07 and 0.19, by up to 13 to 17 points of TPF gain across the four LLaDA2 grids, with exact ties elsewhere, as Figure 6 shows. On SDAR the gains are +8.9 points at $\alpha = 0.12$ on math, and +17.3 points at $\alpha = 0.17$ and +16.6 points at $\alpha = 0.20$ to 0.21 on code. On the Fast-dLLM-v2 grid of Appendix D.8 the same gap reappears at $\alpha = 0.14$ (+9.7 points) and $\alpha = 0.19$ (+22.4 points), and on the grid of 50 configurations of Appendix D.11 at three budgets in each of its two larger nested grids.

Each gain arises when a configuration's p-value lands between Bonferroni's δ/m and Holm's threshold $\delta/(m - i + 1)$ at its rank i, most visibly at Holm's last step at full δ. On LLaDA2 math

Table 5: Intervals for every configuration that Redline deploys, other than the reference. Gain and net accuracy change with 95% paired-bootstrap intervals over prompts, and joint risk $\hat{R}$ with its 95% Clopper–Pearson interval and one-sided 90% upper bound. Budgets that deploy the same configuration share a row.

		gain		net accuracy		joint risk		
grid	α	**value**	**95% interval**	**value**	**95% interval**	$\hat{R}$	**95% interval**	**90% bound**
Block-diffusion serving, TPF gain (%)								
LLaDA2 math	≥ 0.10	$+37.5$	$[34.0, 41.1]$	-4.0	$[-5.9, -2.0]$	0.072	$[0.057, 0.090]$	0.084
LLaDA2 math, large grid	0.10	$+65.4$	$[60.8, 70.2]$	-3.4	$[-5.4, -1.4]$	0.073	$[0.058, 0.091]$	0.085
LLaDA2 math, large grid	≥ 0.15	$+72.0$	$[67.7, 76.3]$	-5.9	$[-8.2, -3.8]$	0.098	$[0.080, 0.118]$	0.111
LLaDA2 code	0.15	$+33.7$	$[26.8, 41.1]$	-7.0	$[-10.1, -3.9]$	0.109	$[0.084, 0.138]$	0.128
LLaDA2 code	0.20	$+42.4$	$[33.1, 52.2]$	-13.1	$[-16.6, -9.8]$	0.157	$[0.127, 0.190]$	0.179
SDAR math	0.10	$+13.1$	$[7.1, 19.6]$	-2.5	$[-4.6, -0.4]$	0.071	$[0.056, 0.089]$	0.083
SDAR math	≥ 0.15	$+36.6$	$[29.2, 44.4]$	-4.6	$[-7.1, -2.3]$	0.105	$[0.087, 0.125]$	0.118
SDAR code	0.15	$+25.7$	$[17.6, 38.9]$	-5.0	$[-7.5, -2.4]$	0.087	$[0.067, 0.111]$	0.103
SDAR code	0.20	$+59.7$	$[38.5, 88.2]$	-14.0	$[-17.5, -10.7]$	0.178	$[0.149, 0.209]$	0.198
Speculative decoding, gain in accepted tokens for each target forward (%)								
Llama GSM8K	0.10	$+13.8$	$[13.2, 14.5]$	-3.3	$[-6.2, -0.6]$	0.072	$[0.051, 0.098]$	0.089
Llama GSM8K	≥ 0.15	$+16.8$	$[16.1, 17.5]$	-9.0	$[-12.3, -5.7]$	0.123	$[0.096, 0.155]$	0.144
Qwen GSM8K	0.05	$+15.0$	$[14.3, 15.7]$	$+0.4$	$[-2.0, +2.7]$	0.033	$[0.019, 0.053]$	0.046
Qwen GSM8K	≥ 0.10	$+18.1$	$[17.4, 18.9]$	-1.6	$[-4.3, +1.2]$	0.057	$[0.038, 0.080]$	0.072
Weight quantization, reduction in weight memory								
Llama GSM8K	≥ 0.10	$2.81\times$	measured	-1.4	$[-4.3, +1.6]$	0.064	$[0.045, 0.089]$	0.081

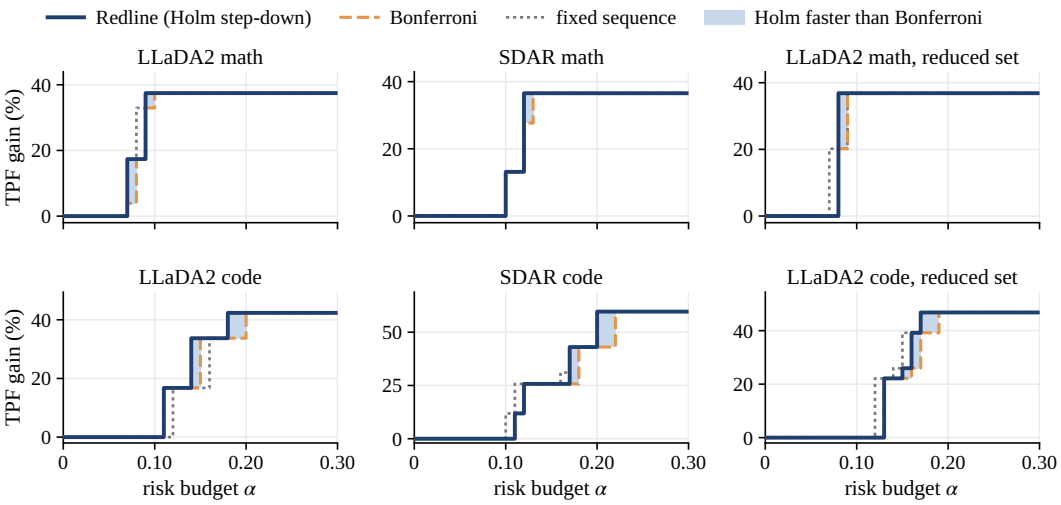


Figure 6: Deployed TPF gain across risk budgets for the six block-diffusion grids under Holm, Bonferroni and fixed-sequence testing. Rows are tasks, and the right column holds the two grids on a reduced prompt set. Shading marks budgets where Holm deploys a faster configuration than Bonferroni.

at $\alpha = 0.07$, its smallest budget with a gain, the acc90/semi90 configuration with $\hat{R} = 0.053$ has $p = 0.0190$, which fails Bonferroni's 0.0143 and passes Holm's 0.025. Holm therefore deploys this configuration at TPF 5.17 ($+17.4\%$), where Bonferroni deploys acc95/semi70 at 4.57 ($+3.9\%$).

Sensitivity to δ. Repeating the test on every grid at $\delta \in \{0.10, 0.05, 0.01\}$ leaves the math configurations deployed at $\alpha = 0.10$ unchanged in both families, LLaDA2 acc85/semi70 at $+37.5\%$ and SDAR acc90/semi90 at $+13.1\%$, down to $\delta = 0.01$. Across the full α grid only 20 of the 360 evaluations over grids, budgets and the three values of δ change the deployed configuration, all at budgets where a configuration sits at the rejection boundary, as expected of an exact test.

Table 6: Smallest math budget with a gain, with the math set subsampled to the code n, over 1,000 uniform or benchmark-stratified draws for each family (Redline, $\delta = 0.10$, fine budget grid). Median, interquartile range, shares of draws under the listed budgets, and the draws at each budget.

family	draw	n	median	IQR	≤ 0.10	< 0.11	≤ 0.11	draws by smallest budget with a gain, 0.05 to 0.12
LLaDA2 math	uniform	542	0.08	[0.07, 0.08]	1.000	1.000	1.000	0, 52, 334, 543, 70, 1, 0, 0
LLaDA2 math	stratified	542	0.08	[0.07, 0.08]	1.000	1.000	1.000	1, 36, 324, 551, 87, 1, 0, 0
SDAR math	uniform	664	0.10	[0.10, 0.11]	0.739	0.739	0.991	0, 0, 0, 14, 224, 501, 252, 9
SDAR math	stratified	664	0.10	[0.09, 0.11]	0.741	0.741	0.994	0, 0, 0, 13, 252, 476, 253, 6

Table 7: Calibration size for the exact binomial test ($\delta = 0.10$). (a) Prompts from which a configuration of true risk R passes with probability at least 0.8 at Holm's first step δ/m. (b) The largest empirical risk that passes at $\alpha = 0.10$ with n prompts.

(a) Prompts for power 0.8

		grids of m configurations		
α	true R	$m = 8$	$m = 16$	$m = 50$
0.02	0.01	1,623	1,828	2,274
0.05	0.04	4,320	5,048	6,217
0.05	0.03	1,008	1,189	1,444
0.10	0.09	8,407	9,842	12,179
0.10	0.08	2,066	2,406	2,966

(b) Margin at $\alpha = 0.10$

	largest $\hat{R}$ (%)		
n	$\delta/8$	$\delta/50$	δ
542	7.0	6.3	8.1
664	7.4	6.6	8.4
1,012	7.8	7.3	8.7
2,000	8.5	8.1	9.1
5,000	9.0	8.8	9.4

Calibration size and the task ordering. At $\alpha = 0.10$ Holm's step passes a configuration only when its empirical risk $\hat{R}$ clears the budget by a finite-sample margin, about 2.1pp at $n = 1{,}012$ and 2.8 to 3.0pp at $n = 542$. At its observed rate but with $n = 1{,}012$ prompts, the LLaDA2 code configuration would be valid at $\alpha = 0.10$ ($p = 0.005$, deployed gain $+16.8\%$), and the smallest budgets with a gain would keep their order, LLaDA2 math at 0.07 against code at 0.10.

The ordering also holds at equal prompt counts. Table 6 draws 1,000 subsets of the 1,012 math prompts at the size of the code set without replacement and re-runs Redline on each over the fine budget grid. The smallest LLaDA2 math budget with a gain has median 0.08 and lies below the code value of 0.11 in every draw, and the SDAR one has median 0.10 and lies at or below 0.11 in 99.1% of draws.

In both families, each of the five configurations faster than the reference fails on a larger share of the reference's correct answers on code than on math, 10.5 to 21.9% against 5.9 to 8.4% for LLaDA2 and 17.8 to 38.9% against 9.2 to 13.5% for SDAR, a ratio between 1.7 and 2.9 at every configuration. Table 7 gives the calibration size from which a configuration one or two points under the budget passes with power at least 0.8 at every larger n, at Holm's first step, the most demanding one.

D.4 Grids on a reduced prompt set

Tables 12 and 13 report two further LLaDA2 grids on a reduced prompt set of the same structure, at most 256 prompts in each subtask, so 256 GSM8K and 256 MATH-500 prompts for math ($n = 512$) and $n = 420$ for code. Their frontiers form the right column of Figure 6, where Holm's gain over Bonferroni peaks at $+16.7$ points at $\alpha = 0.08$ on math and $+13.2$ points at $\alpha = 0.16$ on code. The main text uses the full grids throughout.

D.5 Forwards and verbosity

TPF conflates fewer forwards with longer outputs, so the grid tables co-report the forwards and tokens of each request for every configuration. The SDAR math configuration deployed at $\alpha = 0.10$ (acc90/semi90) cuts compute cleanly, with the forwards of each request falling from 147.4 to 123.9 (-15.9%) and tokens by 4.9%, so its $+13.1\%$ TPF understates the saving.

Table 8: Full grid for LLaDA2 math. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	6.050	128.8	779	0.072 (73/1012)	0.999	**<u>0.001</u>**	**<u>3e-14</u>**	**<u>5e-30</u>**
acc85/semi90	5.850	134.3	786	0.066 (67/1012)	0.990	**1e-4**	**1e-16**	**4e-33**
acc90/semi70	5.456	144.4	788	0.064 (65/1012)	0.981	**4e-5**	**2e-17**	**3e-34**
acc90/semi90	5.165	149.5	772	0.053 (54/1012)	0.718	**6e-8**	**2e-22**	**7e-41**
acc95/semi70	4.571	171.4	784	0.051 (52/1012)	0.616	**1e-8**	**2e-23**	**3e-42**
acc95/semi90 (reference)	4.401	166.3	732	0.000 (0/1012)	**<u>3e-23</u>**	**5e-47**	**4e-72**	**8e-99**
acc99/semi90	3.287	203.4	668	0.040 (40/1012)	0.069	**5e-13**	**3e-30**	**8e-51**

D.6 Oracle gates under matched accounting

The oracle of Figure 2c uses the same FLOP-weighted accounting and the same replay as the static rule and the learned gate. We report two oracle notions. **Unconstrained ceiling.** Every zero-commit forward is removed at no scheduling cost, the upper bound for any gate that leaves commits unchanged. It saves 37.31% FLOP on LLaDA2 and 38.31% on SDAR, $\mathbf{+0.98}$**pp and** $\mathbf{+1.30}$**pp** over the static rule.

Schedulable oracle. It skips exactly the zero-commit forwards, replayed closed loop, and pays for a forced forward whenever every active block would skip. It saves 36.26% and 37.36%, $\mathbf{-0.07}$**pp and** $\mathbf{+0.35}$**pp** against the static rule, which numerically exceeds it on LLaDA2. Both schedules drop a block's leading zero-commit forwards for free. The static rule still pays for its unskipped zeros, those of the oldest block and those inside a block after its first commit, and the oracle pays for its forced steps, which form the larger total on LLaDA2.

Even with perfect knowledge of every forward, then, no realizable gate clears the static rule by more than 0.35pp in either family under the accounting that the throughput regime uses.

D.7 A three-dimensional grid

The block-add schedule enters as a third tested dimension (accept $\times$ semi $\times$ add, 9 configurations, $n = 1{,}012$). Redline deploys the same acc85/semi70 configuration at $+37.5\%$ TPF at $\alpha = 0.10$, which reproduces the two-dimensional result exactly, and the add axis is never selected, since both of its directions buy about 4.5 to 4.7pp of risk for at most $+0.7\%$ TPF. Redline needs no ordering of the thresholds in three dimensions either.

D.8 An external engine

Every speedup below is reported with its α at $\delta = 0.10$. **Fast-dLLM-v2** (Qwen, GSM8K, $n = 512$) yields a deployed frontier over a five-configuration grid of accept thresholds, with deployed gains of $+15.3\%$ TPF at $\alpha = 0.10$ ($\hat{R} = 0.070$), $+25.0\%$ at $\alpha = 0.15$ and $+47.4\%$ at $\alpha = 0.20$.

D.9 The mean-accuracy rule on the same splits

We run the mean-accuracy rule on the four main grids under the split protocol of Appendix A.7, with $K = 1{,}000$ random splits for each grid at calibration fraction 0.5 and, as checks, 0.3 and 0.7. Three selectors see only the calibration half and are scored only on the test half. Redline is the exact binomial test with Holm at $\delta = 0.10$, deploying the fastest valid configuration, or the reference when none is valid. MEAN(t) deploys the fastest configuration whose calibration-half accuracy is within t points of the reference's calibration-half accuracy, for $t \in \{0, 1, \ldots, 8, 10\}$, and every tolerance is reported. REF always deploys the reference.

Exceedance is the fraction of splits whose deployed configuration has test-half $R > \alpha$. Test-half R is itself an estimate, so a configuration whose population risk sits at the budget crosses it on the test half

Table 9: Full grid for LLaDA2 code. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	7.566	28.9	219	0.157 (85/542)	1.000	1.000	0.697	**0.006**
acc85/semi90	7.107	28.6	203	0.109 (59/542)	1.000	0.778	**0.003**	**9e-9**
acc90/semi70	6.793	30.4	207	0.138 (75/542)	1.000	0.998	0.245	**1e-4**
acc90/semi90	6.206	34.0	211	0.076 (41/542)	0.996	0.031	**1e-7**	**7e-16**
acc95/semi70	5.495	38.3	210	0.100 (54/542)	1.000	0.525	**4e-4**	**2e-10**
acc95/semi90 (reference)	5.315	38.9	207	0.000 (0/542)	**8e-13**	**2e-25**	**6e-39**	**3e-53**
acc99/semi70	3.977	53.7	213	0.089 (48/542)	1.000	0.209	**1e-5**	**1e-12**
acc99/semi90	3.691	53.3	197	0.035 (19/542)	0.062	**9e-9**	**2e-18**	**7e-30**

Table 10: Full grid for SDAR math. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	5.263	102.9	542	0.105 (106/1012)	1.000	0.714	**2e-5**	**2e-16**
acc90/semi70	4.956	115.0	570	0.099 (100/1012)	1.000	0.476	**1e-6**	**2e-18**
acc85/semi90	4.920	112.0	551	0.093 (94/1012)	1.000	0.244	**4e-8**	**1e-20**
acc90/semi90	4.361	123.9	540	0.071 (72/1012)	0.999	**9e-4**	**1e-14**	**2e-30**
acc95/semi70	4.296	136.0	584	0.083 (84/1012)	1.000	0.037	**9e-11**	**8e-25**
acc95/semi90 (reference)	3.854	147.4	568	0.000 (0/1012)	**3e-23**	**5e-47**	**4e-72**	**8e-99**
acc99/semi70	3.092	178.8	553	0.084 (85/1012)	1.000	0.047	**2e-10**	**2e-24**
acc99/semi90	2.891	183.8	531	0.038 (38/1012)	0.036	**6e-14**	**2e-31**	**2e-52**

about half the time, which inflates exceedance for every selector alike. The selectors are therefore compared with each other on the same splits at matched speed. The mean rule is also handed the reference's own calibration accuracy, which is more than a leaderboard reader has.

Table 14 prints, for each grid and budget, the mean TPF gain and exceedance of Redline beside the two tolerances that bracket its speed and beside $t = 10$. At matched speed the mean rule exceeds the budget far more often wherever the grid holds configurations near it (SDAR math and LLaDA2 code at $\alpha = 0.10$, SDAR code at 0.15 and both math grids at 0.05). The two rules agree where every faster configuration is well under the budget (LLaDA2 math at $\alpha \geq 0.10$, both math grids at $\alpha \geq 0.15$ and LLaDA2 code at 0.15), and no tolerance reaches the speed of Redline at $\alpha = 0.20$ on either code grid. A calibration fraction of 0.3 leaves every reading unchanged, and at 0.7 the SDAR math reading holds (1.6% against 13.7 to 37.5% at $\alpha = 0.10$). At both fractions, as at half, no tolerance is at once as fast as Redline on LLaDA2 math and as rarely over the budget on SDAR math at $\alpha = 0.10$.

At every printed budget, each tolerance tested, from 0 to 10 points, either gains less TPF than Redline on some grid or exceeds the budget on some grid in at least three times as many of the $1{,}000$ held-out splits and in at least 40 more of them. At $\alpha = 0.10$, for instance, the tolerances up to four points fall at least 3.4 points of TPF gain short of Redline on LLaDA2 math, and those from five points up exceed the budget on SDAR math in 71.1 to 72.4% of held-out splits and in 65.3 to 100% of splits against the pooled risk, against 3.4% and 0.0% for Redline.

Stronger baselines on the same splits. Table 15 runs every selector a reader might prefer on the same $1{,}000$ half splits and reports exceedance under both measures, the test-half risk of the deployed configuration and its risk pooled over all n prompts, which the calibration half also informs and which a configuration selected on that half therefore reads optimistically. For each grid and budget, it gives the mean TPF gain of the deployed configuration, counting deployments of the reference as zero, and both exceedances. Its columns are Redline, Bonferroni (BONF), the fixed-sequence walk (FIXSEQ),

Table 11: Full grid for SDAR code. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	4.253	17.2	73	0.178 (118/664)	1.000	1.000	0.978	**0.081**
acc85/semi90	3.810	18.0	69	0.145 (96/664)	1.000	1.000	0.372	**1e-4**
acc90/semi70	3.492	19.4	68	0.139 (92/664)	1.000	0.999	0.222	**2e-5**
acc90/semi90	3.349	20.6	69	0.087 (58/664)	1.000	0.153	**9e-7**	**1e-15**
acc95/semi70	2.980	23.7	71	0.081 (54/664)	1.000	0.059	**7e-8**	**3e-17**
acc95/semi90 (reference)	2.664	24.2	64	0.000 (0/664)	**2e-15**	**4e-31**	**1e-47**	**4e-65**
acc99/semi90	2.299	32.8	75	0.018 (12/664)	**2e-5**	**2e-17**	**2e-31**	**4e-47**
acc99/semi70	2.225	30.7	68	0.081 (54/664)	1.000	0.059	**7e-8**	**3e-17**

Table 12: Full grid for LLaDA2 math on the reduced prompt set. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	5.973	131.7	787	0.057 (29/512)	0.789	**3e-4**	**3e-11**	**2e-20**
acc85/semi90	5.850	129.8	759	0.070 (36/512)	0.983	**0.012**	**2e-8**	**2e-16**
acc90/semi70	5.317	146.2	778	0.064 (33/512)	0.941	**0.003**	**2e-9**	**5e-18**
acc90/semi90	5.245	150.9	792	0.053 (27/512)	0.659	**8e-5**	**3e-12**	**1e-21**
acc95/semi70	4.534	165.4	750	0.051 (26/512)	0.584	**4e-5**	**1e-12**	**2e-22**
acc95/semi90 (reference)	4.364	159.2	695	0.000 (0/512)	**4e-12**	**4e-24**	**7e-37**	**2e-50**
acc99/semi70	3.442	207.7	715	0.049 (25/512)	0.504	**2e-5**	**3e-13**	**5e-23**
acc99/semi90	3.287	201.8	663	0.039 (20/512)	0.150	**2e-7**	**3e-16**	**1e-26**

the uncorrected test (UNCORR), the plug-in rule (PLUGIN), the net-drop test NETHOLM with a Hoeffding (H), Hoeffding–Bentkus (HB) or betting (WSR) p-value, the mean rule with a normal bound on the drop (MEANCI), and MEAN(t) at the two tolerances that bracket the speed of Redline in Table 14 and at $t = 10$. Where no tolerance from 0 to 10 runs slower than Redline, or none runs as fast, the bracket does not exist and the table says so.

Bonferroni and the fixed-sequence walk in the order of the design thresholds, the two other corrections with a family-wise guarantee, exceed the budget in at most 1.6% of splits against the pooled risk, and Bonferroni deploys a slower configuration than Redline wherever the two differ. Dropping the multiplicity correction, the uncorrected exact test at δ for each configuration, raises held-out exceedance to 17.5% on SDAR math at $\alpha = 0.10$ and 18.7% on SDAR code at 0.15. Dropping the margin as well, the plug-in rule $\hat{R} \le \alpha$ on the calibration half exceeds the budget in 28.5 to 58.4% of splits pooled and 28.9 to 64.1% held out on LLaDA2 math at 0.05, LLaDA2 code at 0.10 and 0.15 and SDAR math at 0.10.

Two selectors that bound the net accuracy drop rather than the joint risk, Holm at δ with the betting p-value of Waudby-Smith & Ramdas (2024) on the rescaled drop and the mean rule with a one-sided 90% normal bound on the drop, keep their own quantity within budget in all but at most 2.5% of splits and still exceed the joint budget in 91.4% and 99.1% of splits on SDAR math at 0.10, the gap between the two risks that our guarantee closes. The Hoeffding and Hoeffding–Bentkus variants (Bates et al., 2021) of that test almost always deploy the reference, since those bounds are loose for a three-valued loss.

Table 13: Full grid for LLaDA2 code on the reduced prompt set. TPF, forwards and tokens of each request, joint risk $\hat{R}\,(k/n)$ and exact binomial p-values at four budgets. Bold marks the configurations valid under Holm, and underlining the one Redline deploys.

					p-value at budget α			
configuration	**TPF**	**forwards**	**tokens**	$\hat{R}\,(k/n)$	**0.05**	**0.10**	**0.15**	**0.20**
acc85/semi70	7.619	28.6	218	0.143 (60/420)	1.000	0.998	0.372	**<u>0.001</u>**
acc85/semi90	7.224	31.8	230	0.121 (51/420)	1.000	0.936	0.055	**1e-5**
acc90/semi70	6.536	32.7	214	0.117 (49/420)	1.000	0.887	**<u>0.030</u>**	**4e-6**
acc90/semi90	6.339	34.2	217	0.090 (38/420)	1.000	0.290	**2e-4**	**7e-10**
acc95/semi70	5.492	40.1	220	0.093 (39/420)	1.000	0.349	**3e-4**	**2e-9**
acc95/semi90 (reference)	5.190	39.6	205	0.000 (0/420)	**<u>4e-10</u>**	**<u>6e-20</u>**	**2e-30**	**2e-41**
acc99/semi70	4.182	54.7	229	0.100 (42/420)	1.000	0.541	**0.002**	**2e-8**
acc99/semi90	3.732	53.5	200	0.043 (18/420)	0.296	**1e-5**	**1e-12**	**6e-21**

Table 14: Redline against the mean-accuracy rule MEAN(t) on 1,000 random half splits. Mean TPF gain, exceedance (splits with test-half $R > \alpha$) and deployments of the reference (ref.), all in percent, for the tolerances that bracket the gain of Redline and for $t = 10$.

	Redline			**MEAN(t) just slower**			**MEAN(t) at or faster**			**MEAN(10)**	
α	**gain**	**exc.**	**ref.**	t	**gain**	**exc.**	t	**gain**	**exc.**	**gain**	**exc.**
					LLaDA2 math						
0.05	0.0	0.1	99.9		none slower		0	0.6	5.7	37.5	99.6
0.10	36.6	0.1	0.0	5	36.6	0.1	6	37.3	0.1	37.5	0.1
0.15	37.5	0.0	0.0	6	37.3	0.0	7	37.5	0.0	37.5	0.0
0.20	37.5	0.0	0.0	6	37.3	0.0	7	37.5	0.0	37.5	0.0
					LLaDA2 code						
0.05	0.0	0.0	100.0		none slower		0	0.0	0.1	33.6	99.9
0.10	1.5	1.3	91.1	2	1.0	2.9	3	3.7	10.2	33.6	73.3
0.15	24.7	0.4	0.1	7	23.9	0.5	8	29.1	1.2	33.6	3.6
0.20	40.2	0.1	0.0	10	33.6	0.1		none as fast		33.6	0.1
					SDAR math						
0.05	0.0	0.0	100.0		none slower		0	1.0	7.0	36.6	100.0
0.10	8.3	3.4	39.5	1	4.5	6.4	2	12.0	21.1	36.6	71.1
0.15	36.5	0.0	0.0	7	36.3	0.0	8	36.5	0.0	36.6	0.0
0.20	36.6	0.0	0.0	8	36.5	0.0	10	36.6	0.0	36.6	0.0
					SDAR code						
0.05	0.0	0.0	100.0		none slower		0	0.0	0.0	36.5	100.0
0.10	1.3	5.8	92.2	2	0.3	1.2	3	1.6	4.9	36.5	70.3
0.15	25.8	4.1	0.0	7	25.6	7.3	8	28.3	18.3	36.5	36.6
0.20	46.9	6.8	0.0	10	36.5	1.3		none as fast		36.5	1.3

D.10 Held-out re-measurement of the deployed configurations

Table 2 reports the TPF gain and net accuracy change of each deployed configuration on the calibration prompts that selected it. Under the split protocol of Appendix A.7, with $K = 1{,}000$ half splits, Redline runs on one half and the deployed configuration's TPF gain, net accuracy change and joint risk are measured on the other half. The tokens and forwards of each prompt come from the decode trajectories. Table 16 prints, for each grid and budget, the full-sample deployment and its in-sample numbers beside the held-out mean and the 2.5 to 97.5 percentile range over the splits that deploy that same configuration, together with the number of splits that deploy the reference.

At $\alpha = 0.10$ the LLaDA2 math configuration is deployed in 894 of 1,000 splits and reads $+37.6\%$ TPF $[+34.1, +41.3]$ and -4.2pp $[-5.9, -2.6]$ held out, against $+37.5\%$ and -4.0pp in sample. The SDAR math configuration is deployed in 548 splits and reads $+13.4\%$ $[+7.6, +19.7]$ and -3.1pp $[-4.7, -1.5]$ against $+13.1\%$ and -2.5pp. Every in-sample TPF gain lies inside its held-out range.

Table 15: Every selector on the same 1,000 random half splits ($\delta = 0.10$). Mean TPF gain of the deployed configuration, and exceedance on the test half and against the pooled risk, in percent.

							NETHOLM				MEAN(t)		
α	measure	Redline	BONF	FIXSEQ	UNCORR	PLUGIN	H	HB	WSR	MEANCI	slower	faster	$t = 10$
LLaDA2 math ($n = 1{,}012$)													
0.05	gain	0.0	0.0	0.1	0.2	7.8	0.0	0.0	8.6	27.0		0.6	37.5
	test	0.1	0.1	1.6	2.4	58.4	0.0	0.0	47.9	89.6	*none*	5.7	99.6
	pool	0.1	0.1	1.6	2.4	58.4	0.0	0.0	50.3	99.5		5.7	100.0
0.10	gain	36.6	32.7	36.7	37.0	37.5	0.0	0.0	37.5	37.5	36.6	37.3	37.5
	test	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	0.1	0.1
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.15	gain	37.5	37.5	37.5	37.5	37.5	29.1	37.3	37.5	37.5	37.3	37.5	37.5
	test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.20	gain	37.5	37.5	37.5	37.5	37.5	37.5	37.5	37.5	37.5	37.3	37.5	37.5
	test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LLaDA2 code ($n = 542$)													
0.05	gain	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	1.8		0.0	33.6
	test	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	12.1	*none*	0.1	99.9
	pool	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	12.1		0.1	100.0
0.10	gain	1.5	1.5	0.4	8.2	21.5	0.0	0.0	9.4	25.5	1.0	3.7	33.6
	test	1.3	1.3	1.0	2.2	28.9	0.0	0.0	11.9	52.2	2.9	10.2	73.3
	pool	0.0	0.0	0.0	0.4	28.5	0.0	0.0	5.3	52.4	0.0	0.7	97.8
0.15	gain	24.7	21.5	17.7	30.4	36.3	0.0	0.0	32.7	35.7	23.9	29.1	33.6
	test	0.4	0.2	1.0	1.1	30.6	0.0	0.0	9.5	23.7	0.5	1.2	3.6
	pool	0.2	0.0	0.2	0.3	30.6	0.0	0.0	9.6	23.7	0.0	0.0	3.6
0.20	gain	40.2	35.8	40.0	40.4	42.3	2.6	12.0	41.3	42.2	33.6		33.6
	test	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	*none*	0.1
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0
SDAR math ($n = 1{,}012$)													
0.05	gain	0.0	0.0	0.0	0.0	0.1	0.0	0.0	2.5	18.2		1.0	36.6
	test	0.0	0.0	0.0	0.0	0.8	0.0	0.0	16.7	88.9	*none*	7.0	100.0
	pool	0.0	0.0	0.0	0.0	0.8	0.0	0.0	16.7	88.9		7.0	100.0
0.10	gain	8.3	6.7	6.2	15.1	29.2	0.0	0.1	35.7	36.5	4.5	12.0	36.6
	test	3.4	2.4	2.8	17.5	64.1	0.0	0.3	71.6	71.4	6.4	21.1	71.1
	pool	0.0	0.0	0.0	1.2	30.4	0.0	0.0	91.4	99.1	0.2	2.6	100.0
0.15	gain	36.5	35.2	36.5	36.5	36.6	18.8	34.0	36.6	36.6	36.3	36.5	36.6
	test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.20	gain	36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.5	36.6	36.6
	test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SDAR code ($n = 664$)													
0.05	gain	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2		0.0	36.5
	test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	5.0	*none*	0.0	100.0
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	5.0		0.0	100.0
0.10	gain	1.3	1.2	5.1	7.1	23.9	0.0	0.0	12.2	26.2	0.3	1.6	36.5
	test	5.8	5.8	8.4	13.6	10.9	0.0	0.0	12.3	14.7	1.2	4.9	70.3
	pool	0.0	0.0	0.0	0.0	0.3	0.0	0.0	1.1	7.5	0.0	0.0	70.1
0.15	gain	25.8	25.3	27.3	27.7	38.0	0.0	0.1	34.5	43.8	25.6	28.3	36.5
	test	4.1	1.5	16.2	18.7	37.4	0.0	0.0	34.5	36.5	7.3	18.3	36.6
	pool	0.0	0.0	0.0	0.0	2.5	0.0	0.0	7.2	11.8	0.1	0.1	1.4
0.20	gain	46.9	38.0	47.7	47.8	58.5	11.4	23.9	56.0	58.8	36.5		36.5
	test	6.8	2.5	6.8	6.8	6.8	0.1	0.6	6.8	6.8	1.3	*none*	1.3
	pool	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0

Table 16: Held-out re-measurement over 1,000 random half splits of the configuration that Redline deploys on one half, measured on the other. Splits deploying the reference and the full-sample configuration, and held-out mean with 95% percentile range, gain in percent and net accuracy in points.

		in sample		splits		held out	
α	deployed	gain	net	ref.	same	gain	net
				LLaDA2 math			
0.05	reference	+0.0	0.0	999	999	+0.0	0.0
0.10	acc85/semi70	+37.5	−4.0	0	894	+37.6 [+34.1, +41.3]	−4.2 [−5.9, −2.6]
0.15	acc85/semi70	+37.5	−4.0	0	1000	+37.6 [+34.1, +41.4]	−4.0 [−5.9, −2.2]
0.20	acc85/semi70	+37.5	−4.0	0	1000	+37.6 [+34.1, +41.4]	−4.0 [−5.9, −2.2]
				LLaDA2 code			
0.05	reference	+0.0	0.0	1000	1000	+0.0	0.0
0.10	reference	+0.0	0.0	911	911	+0.0	0.0
0.15	acc85/semi90	+33.7	−7.0	1	465	+33.8 [+26.6, +41.3]	−8.2 [−10.3, −5.9]
0.20	acc85/semi70	+42.4	−13.1	0	755	+42.7 [+33.2, +52.9]	−13.8 [−16.6, −11.4]
				SDAR math			
0.05	reference	+0.0	0.0	1000	1000	+0.0	0.0
0.10	acc90/semi90	+13.1	−2.5	395	548	+13.4 [+7.6, +19.7]	−3.1 [−4.7, −1.5]
0.15	acc85/semi70	+36.6	−4.6	0	994	+36.6 [+29.1, +44.5]	−4.7 [−7.3, −2.2]
0.20	acc85/semi70	+36.6	−4.6	0	1000	+36.6 [+29.1, +44.5]	−4.7 [−7.3, −2.2]
				SDAR code			
0.05	reference	+0.0	0.0	1000	1000	+0.0	0.0
0.10	reference	+0.0	0.0	922	922	+0.0	0.0
0.15	acc90/semi90	+25.7	−5.0	0	938	+25.5 [+17.4, +33.8]	−5.0 [−7.5, −2.4]
0.20	acc85/semi70	+59.7	−14.0	0	311	+58.8 [+37.8, +79.8]	−15.8 [−18.1, −13.9]

D.11 A LLaDA2 MATH GRID OF 50 CONFIGURATIONS

The LLaDA2 math grid of Table 2 tests $m = 7$ configurations, Holm's correction is cheap at that size, and from $\alpha = 0.10$ on its deployed configuration is the most aggressive one of the grid, so the grid itself bounds the attainable gain. To measure what a larger grid costs and returns, we served a grid of 50 configurations over the engine's three lossy thresholds, accept $\{0.75, 0.80, 0.85, 0.90, 0.95\} \times$ semi-completion $\{0.5, 0.6, 0.7, 0.8, 0.9\} \times$ add-block $\{0.10, 0.30\}$, on the same math set, scorer and serving batch ($n = 1{,}012$). Every configuration, the reference acc95/semi90/add0.10 included, was served anew for this grid, so every risk and gain below is relative to that reference. The add 0.30 configurations run within 5% TPF of their add 0.10 partners at every accept and semi-completion setting (ratios 0.973 to 1.023), so the second add level doubles the grid with near-duplicate hypotheses, which stresses the multiplicity correction.

Table 17 runs Redline at $\delta = 0.10$ on three nested grids of the same served outputs, the six configurations the grid shares with the main grid (accept $\{0.85, 0.90, 0.95\} \times$ semi-completion $\{0.7, 0.9\}$ at add 0.10, $m = 6$), the add 0.10 slice ($m = 25$) and the whole grid ($m = 50$). The grid of six deploys the same configuration as the main grid, acc85/semi70, at every budget from 0.10 on, at +33.4% against the reference of this grid.

On the whole grid the configuration deployed at $\alpha = 0.10$ moves to acc75/semi80/add0.30 at +65.4% ($k = 74$, 36 of 50 configurations valid), and at $\alpha = 0.15$ and 0.20 to acc75/semi50/add0.10 at +72.0% with all 50 valid, with net accuracy changes of −3.4 and −5.9pp beside the risk bound as in Table 2. On the add 0.10 slice alone the configuration deployed at $\alpha = 0.10$ is acc80/semi50 at +54.7% (19 of 25 valid), so at this budget the step from 25 to 50 hypotheses added a faster valid configuration rather than removing one.

The price of the correction. Holm and Bonferroni deploy the same configuration at every printed budget except $\alpha = 0.10$ in the grid of 25. At $\alpha = 0.10$ on the whole grid the deployed configuration's $p = 0.0018$ sits under Bonferroni's $\delta/50 = 0.0020$ and under Holm's threshold at its rank, 0.0056. On the fine grid of 30 budgets, the configuration deployed by Holm is faster than Bonferroni's at three budgets in the grid of 50 ($\alpha = 0.09$, 0.11 and 0.12, by 9.1, 3.0 and 3.1 points of TPF gain), at three in the grid of 25 (0.10 to 0.12, at most 6.1 points) and at none in the grid of six.

Table 17: The LLaDA2 math grid of 50 configurations tested by Redline as three nested grids of the same served outputs ($n = 1{,}012$, $\delta = 0.10$). For each grid and budget, the deployed configuration, the number of valid configurations, its violations k, joint risk, TPF, gain and net accuracy change.

α	deployed	valid	k	$\hat{R}$	TPF	TPF gain (%)	net accuracy (pp)
The six configurations shared with the main grid, $m = 6$							
0.05	reference	1 of 6	0	0.000	4.42	+0.0	+0.0
0.10	acc85/semi70/add0.10	6 of 6	53	0.052	5.90	+33.4	−1.5
0.15	acc85/semi70/add0.10	6 of 6	53	0.052	5.90	+33.4	−1.5
0.20	acc85/semi70/add0.10	6 of 6	53	0.052	5.90	+33.4	−1.5
The add 0.10 slice, $m = 25$							
0.05	reference	1 of 25	0	0.000	4.42	+0.0	+0.0
0.10	acc80/semi50/add0.10	19 of 25	79	0.078	6.84	+54.7	−4.0
0.15	acc75/semi50/add0.10	25 of 25	99	0.098	7.60	+72.0	−5.9
0.20	acc75/semi50/add0.10	25 of 25	99	0.098	7.60	+72.0	−5.9
The whole grid, $m = 50$							
0.05	reference	1 of 50	0	0.000	4.42	+0.0	+0.0
0.10	acc75/semi80/add0.30	36 of 50	74	0.073	7.31	+65.4	−3.4
0.15	acc75/semi50/add0.10	50 of 50	99	0.098	7.60	+72.0	−5.9
0.20	acc75/semi50/add0.10	50 of 50	99	0.098	7.60	+72.0	−5.9

Against an uncorrected test of each configuration at $p \le \delta$, which carries no family-wise guarantee, Holm gives up 3.0 points at $\alpha = 0.10$ on the whole grid (that test would deploy acc75/semi70/add0.10 at $+68.4\%$) and nothing at 0.15 or 0.20. A grid seven times larger therefore costs at most 3.0 points of gain at the printed budgets and reaches configurations the small grid does not contain.

Held out. Under the split protocol of Appendix D.10 on the grid of 50, the deployed configuration's test-half risk exceeds the budget in 5.3% of splits at $\alpha = 0.10$ and in no split at 0.15 and 0.20, where every split deploys the same configuration and reads $+72.0\%$ $[+67.9, +76.3]$ held out.

Grid size. For Figure 5 we test 1,000 random sub-grids of 8, 16 and 32 configurations of the grid of 50, each holding the grid's reference and $m - 1$ of its other configurations, with Redline at $\delta = 0.10$ on all n prompts, and place the designed nested grids of Table 17 beside them. At $\alpha = 0.10$, 0.15 and 0.20 no sub-grid deploys the reference. At $\alpha = 0.10$ the mean deployed gain rises from $+53.3\%$ at $m = 8$ through $+57.0\%$ and $+61.3\%$ to the whole grid's $+65.4\%$, and at $\alpha = 0.15$ and 0.20, whose deployed configurations coincide in every sub-grid, from $+65.4\%$ to $+72.0\%$. Each step is larger than its Monte Carlo standard error, which is at most 0.32 points. The designed grid of six reaches $+33.4\%$ at every budget from 0.10, so a larger grid adds valid configurations faster than the correction removes them.

E Speculative Decoding and Quantization

E.1 Typical acceptance

The lossy acceptance rule is the typical acceptance of Medusa (Cai et al., 2024), which accepts a draft token x when

$$p_{\text{target}}(x) \;>\; \min\big(\epsilon,\; a \cdot \exp(-H(p_{\text{target}}))\big), \tag{6}$$

with H the entropy of the target's next-token distribution and the entropy coefficient coupled as $a = \sqrt{\epsilon}$. A lower ϵ accepts more draft tokens, which is faster and lossier, so ϵ is the setting that Redline searches, not a risk level. We apply the published rule on top of greedy verification rather than inside a deployed serving stack. The tested grid is the lossless reference and the six lossy settings $\epsilon \in \{0.4, 0.3, 0.2, 0.1, 0.05, 0.02\}$. Five further settings, $\epsilon \in \{0.8, 0.7, 0.6, 0.5, 0.45\}$, fill in the frontier on the same 512 calibration prompts. Testing all eleven with Holm is strictly more conservative than testing the six, and it gives identical deployments and gains at $\alpha \in \{0.05, 0.10, 0.15, 0.20\}$ on both draft pairs, for instance $\epsilon = 0.1$ at $+13.8\%$ on the Llama pair at $\alpha = 0.10$.

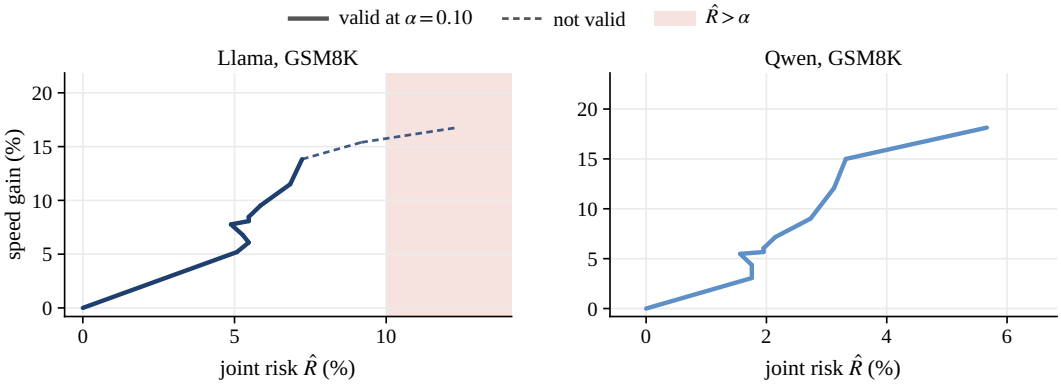


Figure 7: Typical acceptance for two draft pairs on GSM8K. Each vertex is a threshold setting at its empirical joint risk and gain in accepted tokens for each target forward over lossless verification. Solid segments join settings valid at $\alpha = 0.10$ and end at the one Redline deploys.

E.2 The lossless reference

The reference accepts a draft token only when it equals the target's argmax, which reproduces the target's greedy output in exact arithmetic. Its reference-relative risk is therefore zero by construction, and its accepted tokens for each target forward are the speedup that costs nothing (4.14 on the Llama pair, $+314\%$ over autoregressive decoding). Standard speculative decoding is lossless in the same sense, since rejection sampling reproduces the target distribution, so testing it would be vacuous. The guarantee is informative only for a lossy acceptance rule, which trades fidelity for speed.

With each grid at its own δ, the Llama setting deployed on GSM8K at $\alpha = 0.10$ gains $+13.8\%$ accepted tokens for each target forward over lossless verification (net accuracy -3.3pp), and Qwen on GSM8K gains from $\alpha = 0.04$ on, with $+15.0\%$ at $\alpha = 0.05$ (net $+0.4$pp) and $+18.1\%$ at $\alpha = 0.10$. At $\alpha = 0.10$ the deployed settings are $\epsilon = 0.1$ for Llama and $\epsilon = 0.02$ for Qwen. Table 2 lists the deployments of both methods at $\alpha = 0.10$, 0.15 and 0.20, and Figure 7 places every setting by its risk and its gain.

E.3 Joint risk against net accuracy

The joint risk bounds the net accuracy drop from above. The net drop equals $R - G$ with $G = \Pr(\text{reference wrong} \wedge \lambda \text{ right}) \geq 0$, so $R \leq \alpha$ bounds the net drop of every valid configuration. The two pairs split cleanly. All eleven lossy Llama settings lose net accuracy (-1.6 to -9.0pp), while ten of the eleven Qwen settings raise it and only $\epsilon = 0.02$ loses. The Qwen setting deployed at $\alpha = 0.05$ has $\hat{R} = 0.033$ and a net change of $+0.4$pp, because its gain term $G = 0.037$ exceeds its risk. Draft agreement does not explain the split. Token-level agreement between the draft and target argmax, measured in lossless mode with four draft tokens, is higher for Llama (90.5%, 4.14 accepted tokens for each target forward) than for Qwen (88.7%, 3.99), yet Llama carries the higher risk. The operating point that passes depends instead on whether a divergent token changes the final answer, which is exactly what the risk measures.

E.4 Quantization

The quantization grid holds bf16, int8, nf4 and fp4 weights for Llama-3.1-8B on GSM8K ($n = 512$), with bf16 as the reference. At $\alpha = 0.10$ all three quantized configurations are valid (int8 at $\hat{R} = 0.045$ and net $+0.2$pp), and the deployment rule of this grid, the smallest memory among valid configurations, selects nf4. The nf4 configuration ties with fp4 in theoretical size, is listed before it and has the lower empirical risk of the two ($\hat{R} = 0.064$ against 0.078). Measured during cached generation on 20 prompts, nf4 reduces weight memory by $2.81\times$ and peak memory by $2.75\times$, short of the theoretical $4\times$ because the embeddings and the output head stay in bf16.

E.5 Answer extraction

The math scorer reads the instructed answer marker and falls back to the last number of the output. On the Llama pair's generations for the full GSM8K test set (1,319 problems), a strict scorer that accepts only the instructed marker, allowing a currency sign and thousands separators, reads within 0.3pp of this scorer on the lossless reference and within 0.08pp on the deployed configuration.

F Engine and Measurement Details

The serving engine (Jin et al., 2026) keeps a fixed number of block slots for each request. A slot holds a placeholder (mask-filled and not yet admitted), an active block (admitted and being denoised), a committed block awaiting its cache write, or a cached block. The admission threshold τ_{add} turns a placeholder into an active block, the accept and semi-completion thresholds decide which positions an active block resolves in a step, and the commit sweep of Appendix C commits resolved blocks in order. Every forward runs over the whole window of slots, placeholders included.

F.1 Occupancy

The occupancy figures of Section 2.3 separate three step-level quantities on math decode traces (GSM8K and MATH-500). **Mask-holding slots** are the slots of the forward window that contain any mask token, placeholders included. A four-slot buffer reads about 3.9 and an eight-slot buffer about 7.8.

Admitted blocks are the active blocks together with committed blocks awaiting their cache write, which are the blocks in flight. They number about 2.0 at buffers 4 and 8 in both families, and about 1.8 at buffer 2, where the buffer itself caps them.

Partially resolved blocks have between 1 and 31 of their 32 positions masked. They number about 1.1 everywhere, 1.14 on the SDAR trace of the reference configuration and 1.13 on the LLaDA2 trace, with at most two in over 99% of steps. Across buffers 2, 4 and 8 the LLaDA2 mean reads 1.10, 1.13 and 1.13, and the SDAR mean reads 1.11 at buffer 2 against 1.16 at buffer 4.

The buffer sweeps use 128 prompts in each benchmark on the same engine build for both families. Snapshots record the state after the commit sweep, and steps that encode the prompt are excluded throughout.

F.2 No-op accounting

The no-op share of Section 2.3 is a fraction of active block-forwards, the forwards of admitted blocks that are not yet committed. Over 512 requests in each family it is 37.3% of 140,812 active block-forwards on LLaDA2 and 38.3% of 120,287 on SDAR. Together with the 19,906 one-step cache writes of committed blocks, these 261,099 active block-forwards make up the 281,005 admitted block-steps at which Appendix C.3 checks the commit invariant. Placeholder slots are forwarded by the static window and enter neither count, so the share understates the total waste of the window.

F.3 Engine configuration

Every configuration in our grids is a real serving run, and each run archives its fully resolved configuration beside its outputs. The LLaDA2 grids ran at buffer depth 2 and the SDAR grids at buffer depth 4, each on one engine build fixed within its family, and every grid compares configurations of one family, one build and one buffer depth. All configurations of the four main grids share block size 32, admission threshold 0.10, greedy decoding, a budget of 4,096 total tokens, 4,096 new tokens and 1,024 forwards for each request, compiled execution with graph capture, and a batch cap of 8 requests, except that the LLaDA2 code grid uses 16 in all eight of its configurations.

Accept thresholds range over $\{0.85, 0.90, 0.95, 0.99\}$ and semi-completion thresholds over $\{0.70, 0.90\}$, with the engine's default, acc95/semi90, as the reference of each family. The LLaDA2 math grid holds seven of these configurations, without acc99/semi70. Each subtask contributes at most 512 prompts, which binds only for GSM8K, so MATH-500 (500), HumanEval+ (164), MBPP (500) and MBPP+ (378) enter whole.

Table 18: Positioning against related work. Guarantee is a distribution-free finite-sample statement about deployed quality, multiplicity is error control over a space of configurations, mechanism is causal evidence for the operating point, and generality is one procedure across cost measures.

line of work	guarantee	multiplicity	mechanism	generality
BD-LM families and serving engine (BD3-LM, SDAR, LLaDA2.0 and 2.1, MBD-LM)	✗ mean accuracy	✗ hand-picked points	stall documented, unexplained	✗
learned-frontier line (distillation, multi-block visibility, RL shaping, in-place revision, Fast-dLLM and v2, D2F, AdaBlock, dParallel, LoPA, d3LLM, DMax)	✗	✗	✗	one cost each
Fast-dLLM Theorem 1	deterministic, conditional on trusting model confidence, no δ	✗	✗	one threshold
CALM (LTT on AR early exit)	✓ finite-sample, relative to the full model	one threshold	✗	early exit only
LTT and conformal risk control (frameworks)	✓	✓ procedure	not a serving system	
lossy AR acceleration (speculative decoding, Medusa typical acceptance, PTQ)	✗ supply the settings	✗	✗	one cost each
Redline (this work)	✓ distribution-free, finite-sample, family-wise δ	✓ Holm over the serving grid	✓ controlled measurements of all three levers	✓ accept, schedule, speculative decoding, quantization

The SDAR grids need one engine change. The upstream engine fixes the SDAR accept threshold at 0.95, and a small sampler-level change makes it configurable, with no training and no new weights. The default 0.95 stays the reference of that family. The same build carries the instrumentation of the measurements of Section 3, disabled in every grid run.

F.4 TPF AND MEASURED THROUGHPUT

The serving logs of the grid configurations record total time together with end-to-end throughput. The LLaDA2 grids and the SDAR code grid each ran on one host, and the SDAR math grid ran on four, with five of its configurations on one host and one on each of the others. Every host of a family ran the same engine build with the settings fixed by the resolved configurations. On every host that served more than one configuration, TPF orders the configurations exactly as end-to-end throughput does, with Spearman $\rho = 1.0$, and the configuration with the highest TPF is also the fastest.

The reason is the static graph. The wall time of a forward varies across the configurations of a host by 0.8% on the LLaDA2 math host, 3.0% on the LLaDA2 code host, 2.5% on the SDAR code host and 0.4% on the SDAR math host, so throughput is TPF divided by a host-level constant, and TPF ordering is throughput ordering at fixed hardware.

G POSITIONING AGAINST RELATED WORK

Table 18 condenses the positioning of Section 6 along four properties, a distribution-free finite-sample guarantee about deployed quality, error control over a space of configurations rather than one threshold, causal evidence for the operating point, and one statement that carries across cost measures. The nearest prior guarantee deserves a note. Theorem 1 of Fast-dLLM (Wu et al., 2026b) is a deterministic step-level inequality that holds once the model's confidence is trusted, so it guarantees neither that trust nor any configuration space and carries no δ. Redline adds the statistical guarantee that the theorem lacks. Like Redline, CALM (Schuster et al., 2022) measures its risk against a reference, the full model, and for a $0/1$ risk its clipped risk difference coincides with our joint loss.