SPECULATING ON THE PAST: LOSSLESS REUSE OF STALE KV CACHES VIA SELF-SPECULATIVE VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-append edits in agentic LLM serving invalidate the cached suffix, forcing a choice between exact refresh, which delays generation, and approximate reuse, which can change the generated action. STALESPEC instead treats the position-aligned stale cache as a self-speculative proposal: the model decodes from the stale view while the exact post-edit suffix is recomputed in parallel, then commits only after fresh verification. The committed distribution is identical to refreshed-context decoding regardless of proposal quality; staleness affects only acceptance and latency. Across controlled 8K–32K edited turns, STALESPEC reduces committed latency by 3–13%, and under KV-packed prefill–decode serving it improves queue-inclusive throughput by 11–19% across three MoE deployments. A cost-aware gate selects profitable turns, and STALESPEC composes with lossless decode-side acceleration. Approximate stale reuse changes 64–99% of tokens and preserves only 0–19% of next actions, while STALESPEC remains exact.

1 INTRODUCTION

Agentic LLM systems repeatedly replace retrieved documents, update memory, insert delayed tool results, and compact histories. A non-append edit preserves an exact common prefix but invalidates every cached state after the first change. The safe response is to refresh the invalidated suffix before decoding. Approximate cache reuse can avoid that wait, but on our held-out agent-derived edited turns it changes 64–99% of free-running tokens and preserves almost none of the measured next actions (0–19% agreement; Table 3). For action-producing models, this is not merely a small numerical approximation: one changed tool call can alter all subsequent turns.

STALESPEC addresses the regime in which exact refresh is expensive relative to the next generation. This occurs most clearly for a long invalidated suffix followed by an action-producing turn, as in insertion, retrieval-set, and memory updates. Rather than replacing exact refresh, the method overlaps it with self-speculative work and uses a cost-aware gate to select turns with sufficient refresh budget. This regime-aware design preserves exactness across all edit types while concentrating acceleration where edited-context refresh is most expensive.

Large language model inference combines a compute-bound prompt *prefill* with sequential, memory-bound *decode*. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates decode through parallel verification while preserving the target distribution. Prefix caching (Kwon et al., 2023; Zheng et al., 2024; Gim et al., 2024) skips prefill when a request shares a bit-exact cached prefix, and serving systems route requests to maximize such hits (Qin et al., 2025). None of these handles a non-append edit whose surviving suffix has reusable content but stale hidden states.

Key observation. The pre-edit cache is stale for the new context, but it still defines a valid autoregressive distribution. Speculative transport requires a valid proposal, not necessarily a smaller model or a close approximation to the target. Existing self-speculation spatially degrades an up-to-date cache (Sun et al., 2024; Sadhukhan et al., 2025; Yue et al., 2026). We instead speculate along the temporal axis: while the exact suffix refresh runs, the same model decodes from a position-aligned stale view. A fresh verification pass transports that proposal to the exact post-edit distribution. Staleness affects only the accepted prefix and latency; it cannot bias the committed distribution. This also differs

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.

from verified compressed-KV decoding, where the exact full KV already exists and is transferred for verification: after an edit, the correct suffix KV is missing.

The deployment is natural under prefill/decode (PD) disaggregation: a prefill pool refreshes the exact suffix while a decode pool proposes from the stale view. The portable realization materializes the shifted suffix into ordinary paged-KV blocks and runs on source-unmodified vLLM (Kwon et al., 2023). Rotation-on-read (RoR) removes that copy at long context by aliasing resident donor pages and applying relative RoPE correction during attention.

Contributions. (1) Exact temporal reuse after edits. We give one self-speculative transport for insertions, deletions, substitutions, memory updates, and compaction, without training or an auxiliary model, and validate exactness under greedy and stochastic decoding across dense/MoE and GQA/MLA models. **(2) Regime-aware serving acceleration.** Controlled committed gains are 3–13%, KV-packed throughput gains are 11–19%, and a cost-aware gate improves a heterogeneous workload from $1.02\times$ to $1.13\times$, within 0.8% of a cell oracle. The gate uses only the refresh-to-decode cost ratio; it does not predict proposal acceptance. After fresh verification, EAGLE-3 and DFlash accelerate the residual continuation, adding up to 20% and 45%, respectively, over matched exact backends. **(3) Proposal-source and long-context analysis.** A resource-factorized study separates stale-state delivery from proposal computation: among the evaluated smaller, quantized, layer-skipped, scheduling, and trajectory-repair alternatives, none provides a better measured replacement for target self-drafting. RoR aliases rather than copies the shifted stale window, keeps long-context KV occupancy within 1% of B1.

2 METHOD

2.1 SETTING

A turn has edited context $c' = (c'_1, \ldots, c'_L)$, obtained from the previous context c by an insertion, deletion, substitution, or contiguous rewrite. Appends are handled by exact prefix caching and are not our target. Let e be the first differing position and $L_{\text{inv}} = L - e + 1$ the invalidated-suffix length. The retained cache K_{stale} is exact for the shared prefix $c'_{<e} = c_{<e}$ and stale thereafter. Refreshing the invalidated suffix takes reference time T_p; one ordinary decode step takes t_d; and G is the generation horizon. Our main agent replay uses $G = 64$. Longer outputs reduce the refresh-to-decode ratio, which we study with measured turn lengths and a generation-budget sweep.

The same model defines next-token distributions p under fresh cache K_{fresh} and q under an assembled stale view, after the same sampling stack. For $x \sim q$, standard speculative transport (Leviathan et al., 2023; Chen et al., 2023) accepts with probability $\min\{1, p(x)/q(x)\}$ and otherwise samples from the normalized residual $(p - q)_+$. Thus the emitted token follows p exactly for any proposal quality. The acceptance test needs only $q(x)$; stochastic residual sampling additionally needs the full processed distribution q.

Table 1: **Notation used throughout the method and analysis.**

Notation	Meaning
$c,\ c'$	previous and edited contexts
$e,\ L_{\text{inv}}$	first changed position; invalidated-suffix length
$K_{\mathsf{stale}},\ K_{\mathsf{fresh}}$	retained stale cache; refreshed exact cache
$q,\ p$	stale-view proposal; fresh target distribution
$T_p,\ t_d$	suffix-refresh time; ordinary decode-step time
$t_\delta,\ t_v(L, n)$	stale-view assembly; verification cost
$G,\ n,\ M$	generation horizon; proposals; committed tokens
$\widehat{\rho}$	estimated refresh-to-decode cost ratio

2.2 STALE-KV SELF-SPECULATION

1. Refresh and assemble concurrently. At time zero, launch an exact chunked refresh of $c'_{\geq e}$ to build K_{fresh}. In parallel, build the proposal cache from the exact shared-prefix KV, an exact prefill of the edited span Δ, and the surviving stale suffix after positional re-alignment. Let this assembly cost t_δ. The reused suffix is position-correct but content-stale, so the assembled cache defines proposal distribution q and is never served directly. Appendix A.1 gives the assembly algorithm and supported edits.

2. Propose during refresh. After assembly, sample $x_1, x_2, \ldots \sim q$ and retain each processed proposal probability $q_i(x_i)$. Stochastic decoding also retains, or exactly reproduces, the full q_i if rejection requires residual sampling. The optimized greedy path instead checks proposal tokens against the fresh-model argmax and replaces the first mismatch with the fresh token.

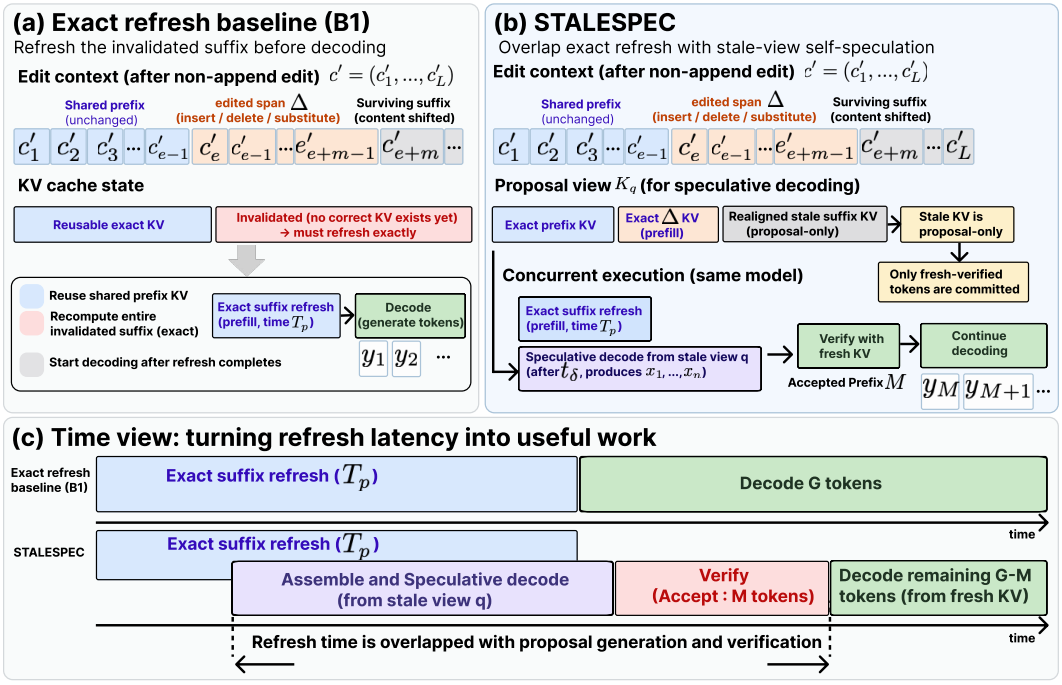

Figure 1: **Overview of StaleSpec.** B1 waits for the invalidated suffix to be refreshed before decoding. StaleSpec instead proposes from an assembled stale view while the exact suffix is refreshed in parallel, then verifies against fresh KV and commits only fresh-verified tokens.

Algorithm 1 StaleSpec for one edited turn (lossless mode).

1: identify the shared prefix, rewritten span, and reusable stale suffix
2: estimate $\widehat{\rho}$; if $\widehat{\rho} < \tau_\rho$, execute B1 and return
3: launch exact suffix refresh and stale-view proposal decoding concurrently
4: draft until refresh completion or the proposal budget
5: retain $q_i(x_i)$ and retain or reproduce full q_i if residual sampling is needed
6: verify all proposals once against fresh KV
7: commit the accepted prefix plus residual/bonus token; continue from fresh KV

The first proposal is available after $t_\delta + t_d$. Proposal tokens append their own stale-view KV, and drafting stops when refresh finishes, EOS appears, or an optional abort rule fires. This changes only the proposal count, not correctness. All proposals remain buffered until fresh verification; no token or external action is released beforehand.

3. Verify once and continue. Once K_{fresh} is ready, one causal parallel forward over n proposals produces fresh distributions p_i and KV at measured cost $t_v(L, n)$. Under stochastic decoding, x_i is accepted with probability $\min\{1, p_i(x_i)/q_i(x_i)\}$. At the first rejection, sample from

$$\frac{(p_i - q_i)_+}{\sum_y (p_i(y) - q_i(y))_+}$$

and discard later proposals; if all n are accepted, draw the standard bonus token. Under greedy decoding, commit the longest prefix matching the fresh argmax and then the fresh argmax at the first mismatch.

Decoding then continues from fresh KV. On accept-all, the optimized verifier continues in place; otherwise a corrected-token continuation starts. Exactness requires the proposal probability in each acceptance test to match the distribution that sampled that token, and the fresh verifier to use the same sampling stack as the reference decoder. Stochastic rejection additionally requires the corresponding full q_i.

Portable materialized realization. Our broadly validated path implements each phase as an ordinary source-unmodified vLLM request with pre-seeded paged-KV block tables. Proposal work runs at low priority and is capped at $2\lceil L_{\text{inv}}/\text{MBT}\rceil + 8$ tokens, where MBT is the chunked-prefill budget. Verification and continuation are fused on accept-all. If an edit shifts the surviving suffix by Δ_{pos}, the path copies that window into fresh pages and applies relative key rotation $R_{\Delta_{\text{pos}}}$; for MLA, only the decoupled positional key is transformed. This path supports the cross-model validation in Sec. 4.

Budget-aware admission. Before drafting, the runtime estimates

$$\widehat{\rho} = \frac{\widehat{T}_p}{G\widehat{t}_d},$$

using the invalidated-suffix length, deployment-local suffix-prefill throughput, and an exponential moving average of decode latency. It enables drafting only when $\widehat{\rho} \geq \tau_\rho$; otherwise it runs exact B1. The gate does not predict proposal acceptance, and both branches are distribution-exact, so a routing error affects only latency. Threshold calibration and causal updates are described in Appendix E.3.

2.3 Rotation-on-read runtime extension

Materializing a shifted window costs $O(S_{\text{win}})$ KV writes per concurrent turn. RoR avoids this copy by giving each draft request a read-only donor view. Within the shifted window, destination position p reads the donor page and slot for $p - \Delta$; outside it, attention uses the ordinary request table. Values are unchanged. For RoPE keys, the relative correction can move to the query:

$$(R_m q)^\top (R_\Delta R_n k) = (R_{m-\Delta} q)^\top (R_n k).$$

The kernel therefore computes $q_{\text{win}} = R_{-\Delta} q$ for window columns while leaving donor keys untouched. Only one boundary block, generation pages, and per-request metadata are newly allocated; prefix and suffix pages remain refcounted aliases.

RoR is an optional runtime optimization, not part of the lossless guarantee. The verifier still reads freshly refreshed KV, so draft-side numerical differences can change acceptance but not the committed distribution. All RoR results compare fixed materialized and fixed RoR modes. Appendix E.5 gives the page mapping, kernel patches, exactness probes, and memory–concurrency results.

3 Correctness and Latency Analysis

All proofs appear in Appendix F.

Operating model. Fix a turn with reference invalidated-suffix refresh time T_p, effective per-token decode latency t_d, configured generation horizon G, and stale-view assembly cost t_δ. Refresh and assembly start concurrently. Let $T_r = \varphi T_p$ be the measured refresh duration while StaleSpec is active. The factor φ includes chunking, scheduling, and interference. Let $\varphi_0 T_p$ be the corresponding duration of the *same suffix-refresh work* for an exact baseline that performs no stale proposal work under the same load. Both factors are defined relative to matched suffix refresh; a full-prefill calibration is not a valid estimator of either factor. Low-contention paired cells can have $\varphi_0 \simeq \varphi$, whereas closed-loop saturation can produce $\varphi > \varphi_0$ because StaleSpec executes additional engine passes.

The refresh window accommodates at most

$$n_{\max} = \max\left\{0, \left\lfloor \frac{\varphi T_p - t_\delta}{t_d} \right\rfloor\right\}$$

proposal tokens. Let $0 \leq n \leq n_{\max}$ be the realized count and $t_v = t_v(L, n)$ their causal parallel verification cost. For $n \geq 1$, let $A \in \{0, \ldots, n\}$ be the accepted proposal-prefix length and $M = \min\{A + 1, G\}$ the number of output tokens committed by that verification round, including the residual token after rejection or the bonus token after full acceptance. If $n = 0$, the implementation bypasses verification and sets $M = 0$.

We summarize accepted work by

$$\beta = \frac{\mathbb{E}[M] t_d}{T_p}, \qquad \rho = \frac{T_p}{G t_d}.$$

Here β is committed decode work produced during refresh, expressed in units of reference refresh time, and ρ compares refresh with ordinary decode. Since $M \le \min\{n+1, G\}$, β cannot exceed the available refresh window or total decode work, up to flooring and one-token boundary terms. For intuition only, under independent per-position acceptance α, $\mathbb{E}[M] = \sum_{j=0}^{\min\{n,G-1\}} \alpha^j$; all predictions use measured, potentially structured $\mathbb{E}[M]$ instead.

Proposition 1 (Edit-invariant exactness of stale-cache speculation)**.** *For any edit and degree of staleness, and for any architecture in which the assembled stale state defines a valid autoregressive proposal q, Algorithm 1 emits a committed sequence distributed identically to standard decoding under K_{fresh}, provided that (i) proposal and verifier distributions include the same sampling transformations as the reference decoder, (ii) the sampled-token probability $q_i(x_i)$ is available for each acceptance test, and (iii) under stochastic decoding, the full distribution q_i is available or exactly reproducible whenever residual sampling is required.*

The acceptance ratio requires only the sampled-token probability $q_i(x_i) > 0$ almost surely; the full q_i is required upon rejection to sample from the positive residual. Proposition 1 establishes the key invariant of StaleSpec: once the assembled stale state defines a valid autoregressive proposal, neither edit type nor cache staleness affects the committed distribution. They determine only the accepted proposal prefix and, consequently, latency. This separation lets one exact transport construction cover insertions, deletions, substitutions, memory updates, and history compaction despite their different cache-invalidation patterns. Appendix B.4 audits the emitted distribution after temperature and top-p processing, including full-distribution residual sampling.

Proposal availability versus commitment. For $n \ge 1$, the first proposal is available at $T_{\text{prop}} = t_\delta + t_d$, whereas the first verification-committed token appears at

$$\text{TTFT}_{\text{commit}} = \max\{t_\delta, \varphi T_p\} + t_v.$$

A blocking exact refresh gives $T_p + t_d$ under our convention. We deliberately do not call T_{prop} a user-visible TTFT: it is usable only by side-effect-free, cancelable, or rollback-safe work. A no-retraction interface waits for verification, so committed turn latency is the primary serving metric.

Theorem 1 (Committed end-to-end latency)**.** *For a realization committing M tokens,*

$$T_{\text{StaleSpec}}(M) \le \max\{t_\delta, \varphi T_p\} + t_v + t_d(G - M),$$

and therefore

$$\mathbb{E}[T_{\text{StaleSpec}}] \le \max\{t_\delta, \varphi T_p\} + t_v + t_d\big(G - \mathbb{E}[M]\big).$$

Let B0 be blocking exact refresh, $T_{\text{B0}} = T_p + Gt_d$, and let B1 be the exact baseline under the same load, $T_{\text{B1}} = \varphi_0 T_p + Gt_d$. Under the first-order model, with assembly hidden and t_v neglected relative to T_p, the predicted speedups are

$$\widetilde{S}_{\text{B0}} = \frac{1+\rho}{1+\rho(\varphi-\beta)}, \qquad \widetilde{S}_{\text{B1}} = \frac{1+\varphi_0\rho}{1+\rho(\varphi-\beta)}.$$

Relative to B1, speculation is profitable to first order when

$$\beta > \varphi - \varphi_0 + \frac{t_v}{T_p}.$$

At matched refresh, $\varphi_0 = \varphi$; at $\varphi_0 = \varphi = 1$, both speedups reduce to $(1+\rho)/(1+\rho(1-\beta))$. With one verification round and negligible assembly, verification, flooring, and one-token boundary terms,

$$\widetilde{S} \le 1 + \min\{\rho, 1/\rho\} \le 2,$$

maximized when refresh and ordinary decode are balanced.

Corollary 1 (Proposal replacement criterion)**.** *Consider two proposal paths a and b under matched refresh, verification, generation horizon, and residual-decoding conditions. Let C_m^{crit} denote the incremental critical-path charge attributable to proposal path m, and let M_m be the number of tokens committed by fresh verification. Under the same first-order decode model,*

$$\mathbb{E}[T_b] < \mathbb{E}[T_a] \quad \Longleftrightarrow \quad t_d\big(\mathbb{E}[M_b] - \mathbb{E}[M_a]\big) > \mathbb{E}[C_b^{\text{crit}}] - \mathbb{E}[C_a^{\text{crit}}].$$

Thus reducing raw proposal FLOPs alone is insufficient when the original proposal work is already hidden by refresh; a replacement must preserve enough fresh-verified work to offset any incremental critical-path charge.

The two B1 terms have distinct meanings. β credits verification-committed decode work produced during refresh, whereas $\varphi - \varphi_0$ charges contention introduced by proposal and verification passes. High acceptance alone therefore does not imply a loaded latency improvement: a saturated colocated engine may have $\varphi > \varphi_0$, while a dedicated refresh pool realizes $\varphi \simeq \varphi_0 \simeq 1$. Our experiments characterize the resulting end-to-end latency at selected load points but do not provide a full load sweep of φ/φ_0; serving results should therefore be interpreted as measurements of those operating points rather than a contention-invariant guarantee. Appendix F proves a bounded-downside corollary.

Pre-service admission. The exact profitability condition contains realized committed work β, which is unknown before serving. The dispatcher therefore uses the pre-service cost coordinate $\widehat{\rho} = \widehat{T}_p/(G\widehat{t}_d)$ and enables drafting only when $\widehat{\rho} \geq \tau_\rho$. The invalidated suffix and both rate estimates are available before proposal decoding, so the rule requires no acceptance prediction. A deployment-specific threshold is calibrated from depth sweeps. In our agent-action experiments $G = 64$; for variable-length traffic, the runtime uses the request cap or a deployment-local expected horizon. Longer outputs reduce ρ and make the gate more conservative, while causal outcome-regret updates handle residual variation from proposal quality and contention. Since both dispatched paths are exact, a gate error affects latency but never the committed distribution.

4 EXPERIMENTS

Setup. We first test deployable exact acceleration, then analyze when and why it works. We ask whether STALESPEC adds benefit on top of EAGLE-3 (Li et al., 2025b) and DFlash (Chen et al., 2026), measure committed serving speed and fidelity on held-out agent turns and KV-packed PD serving, and study proposal survival, admission, and long-context scaling.

The core suite contains 318 context–edit instances from software-agent histories, retrieval updates, memory operations, and tool-registry changes. Unless noted, decoding is greedy and the main replay uses $G = 64$, a conservative anchor for agent traffic: across 9,858 assistant turns from 400 real SWE-agent trajectories, the median output is 19 tokens and 82% emit at most 64 tokens. Serving uses NVIDIA B200 GPUs and compares against B1, which reuses the exact prefix but blocks until the invalidated suffix is refreshed. Paired comparisons hold the model, edit, context, hardware, concurrency, and decode backend fixed. Appendix B gives the complete protocols and run counts, and Appendix E.4 reports the full output-length distribution and generation-budget sweep.

Composition with modern lossless decode drafters. A deployment may already accelerate fresh decoding, so we test the incremental gain from adding STALESPEC. On Qwen3-8B with 1:1 PD at $C \in \{8, 16\}$, each cell compares the same backend—ordinary decoding, EAGLE-3, or DFlash—with and without STALESPEC.

Table 2 shows positive committed-p50 gains in all 18 cells: up to $+37\%$ with ordinary decoding, $+20\%$ over EAGLE-3, and $+45\%$ over DFlash. The methods act at different stages: STALESPEC converts suffix-refresh time into a fresh-verified prefix of M tokens, then the decode-side drafter accelerates the remaining $G - M$ fresh tokens. Appendix D.1 gives the matched-backend accounting.

End-to-end agent-turn serving: speed and fidelity. We next replay held-out edited contexts from real agent trajectories on larger MoE and MLA deployments. Committed latency runs from edit arrival to the G-th verified token and includes suffix refresh, stale-view assembly, proposal decoding, verification, continuation, and the measured-bandwidth P→D transfer term. Generated turns are not fed back into later contexts, so this is an end-to-end model-serving benchmark rather than a closed-loop agent-task evaluation.

Experiments use 1:1 PD pools, $C = 8$, greedy $G = 64$, $N = 32$ turns per cell, and three paired runs. We compare B1 with PromptCache (Gim et al., 2024) (verbatim stale reuse) and CacheBlend (Yao et al., 2025) using its HKVD policy: layer-1 stale–fresh deviation selects the top 15% positions for fresh-KV splicing, and we charge the measured selection and recompute costs. Appendix D.4 gives the full implementation and controls.

STALESPEC is the only reuse path that is both verifier-exact and no slower than B1 to commit, while making the first proposal 10–17× earlier. CacheBlend and PromptCache commit faster (1.8–2.8×) but change 64–99% of free-running tokens and preserve only 0–19% of next actions. Their teacher-forced disagreement is much smaller (1.6–2.9%), showing why small local errors can still change a

Table 2: **StaleSpec composes with fresh-decode acceleration.** Incremental committed-p50 latency reduction from adding StaleSpec to a fixed Qwen3-8B backend under 1:1 PD. Drafter-only gains are excluded; all cells remain exact.

Residual decode backend	Benchmark	C=8	C=16
None	SWE-smith	+15%	+8%
	SWE-rebench	+37%	+24%
	TerminalBench	+12%	+26%
EAGLE-3	SWE-smith	+14%	+8%
	SWE-rebench	+20%	+19%
	TerminalBench	+10%	+16%
DFlash	SWE-smith	+33%	+14%
	SWE-rebench	+45%	+32%
	TerminalBench	+13%	+13%

Table 3: **Committed serving on replayed agent-derived edited turns.** Results are relative to refresh-matched B1 under 1:1 PD and $C = 8$ (mean±std over three runs). Speed reports *committed-turn / first-proposal* speedup; fidelity reports *token deviation* ↓ */ next-action agreement* ↑ against the fresh verifier.

		speed vs. B1 (committed / first)			fidelity: deviation / agreement (%)		
Model	Workload	StaleSpec	CacheBlend	PromptCache	StaleSpec	CacheBlend	PromptCache
GLM-4.7-Flash (MLA)	TerminalBench	**1.31±.19× / 14.3±1.3×**	2.44× / 4.9×	2.28× / 10.1×	**0 / 100**	73/8	79/6
	SWE-smith	**1.14±.04× / 12.4±1.6×**	2.76× / 5.1×	2.52× / 10.8×	**0 / 100**	67/12	73/0
	SWE-rebench	**1.11±.10× / 10.3±.8×**	1.95× / 3.7×	1.86× / 7.8×	**0 / 100**	89/3	91/3
Qwen3-235B -A22B (TP4)	TerminalBench	**1.11× / 16.9×**	1.87±.01× / 6.9×	2.25× / 12.2×	**0 / 100**	98/0	93/5
	SWE-smith	**1.15× / 13.8×**	2.22±.06× / 4.0×	2.08× / 6.8×	**0 / 100**	93/0	98/0
	SWE-rebench	**1.18× / 12.8×**	1.79±.02× / 6.4×	2.11× / 10.1×	**0 / 100**	99/0	99/0

free-running agent trajectory. We therefore treat their speed as an approximate-reuse tradeoff, not an exact comparison.

Deployment-shaped KV-packed PD throughput. On LongBench multi-document QA with an early document drop that invalidates roughly 12K tokens, StaleSpec improves queue-inclusive throughput by 11–19% across Qwen3-30B-A3B, Qwen3-235B-A22B (Yang et al., 2025), and GLM-4.7-Flash (Z.ai, 2026) under 1:1 PD and $C = 8$. Concurrent p95 latency improves by 1.36–1.85×, and the first proposal appears 19–22× earlier than B1 (Appendix Table 7). The measured handoff-specific term is only 0.3% of turn latency; Appendix B.5 gives the full timing and occupancy breakdown.

Transport-correctness audit. We revalidate exact transport after every systems optimization. In fp32, committed outputs match direct refreshed-context decoding, and a forced-proposal test confirms that the speculative path is exercised. Under stochastic decoding, emitted-token distributions match fresh-target sampling within Monte Carlo variation; an intentionally mismatched sampling stack increases the discrepancy by about 6× (Appendix B.4). The fully stochastic serving path still reduces committed latency by 8–11% on Qwen3-30B-A3B (Appendix D).

Stochastic serving. On Qwen3-30B-A3B with temperature 0.9 and top-$p = 0.95$, the full stochastic path retains an 8–11% committed-latency reduction over B1; proposal-distribution capture costs only 1.5% of turn latency.

Controlled latency. Across paired 8K–32K edited turns, StaleSpec reduces committed latency by 3–13% relative to matched B1; Appendix C–E reports the full dense-model and long-context sweeps.

Why stale proposals survive. For each trace, we sample autoregressive proposals from the stale view and verify them under the fresh model. At proposal position i, the sampled token is accepted with probability

$$a_i = \min\{1, p_i(x_i)/q_i(x_i)\}.$$

Averaging cumulative acceptance weights over stale-view proposal trajectories gives the prefix-survival probability $S(i) = \Pr(A \geq i)$ and hence the expected accepted-run length $\mathbb{E}[A] = \sum_i S(i)$.

Table 4: **Expected accepted-run length by edit type.** Median per-trace $\mathbb{E}[A]$ with trace-bootstrap 95% CIs; the final column gives the per-token acceptance gain from positional re-alignment.

Edit	Qwen3-8B	Qwen3-32B	Llama-3.1-8B	Align gain
append	α=1	α=1	α=1	–
insert	35.3 [31.9, 42.4]	27.9 [21.9, 34.0]	41.4 [31.9, 49.5]	.03–.05
elide	30.4 [28.4, 34.0]	17.9 [12.6, 25.2]	35.3 [24.3, 48.2]	.21–.28
memory	29.3 [26.4, 45.9]	20.8 [17.6, 24.8]	48.1 [38.0, 61.6]	.10–.13
swap	22.7 [20.0, 24.6]	22.1 [17.6, 25.9]	12.2 [8.3, 16.8]	.21–.22
reorder	20.0 [15.7, 25.4]	22.6 [18.3, 24.6]	16.2 [12.3, 17.6]	≈ 0
compaction	5.5 [4.2, 7.4]	5.9 [4.4, 7.2]	8.1 [6.0, 9.8]	.35–.80

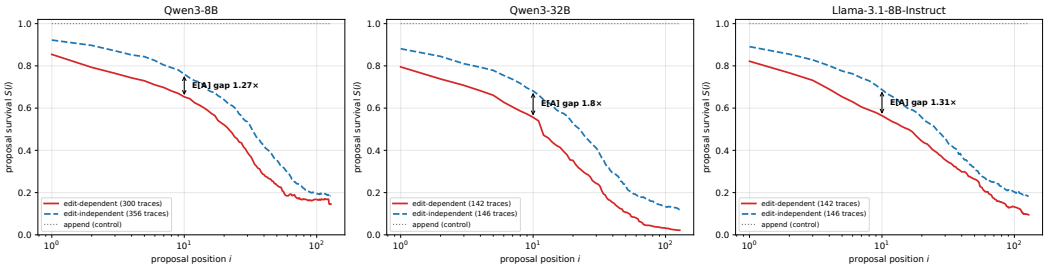

Figure 2: **Proposal survival is semantically structured.** $S(i) = \Pr(A \geq i)$ is estimated under the actual stale-view proposal process. Survival decays earlier when generation depends on the edited document. Append is exact because it does not invalidate the cached prefix.

Accepted runs are longer when the continuation does not yet depend on the edited content and shorter when it does.

Semantic dependence matters more than edit size alone. For document swaps, proposals fail earlier when the continuation needs the changed document than when it does not, consistently across all three models (Fig. 2). Positional re-alignment is also important, improving per-token acceptance by up to 0.80.

Cost-aware operating region. Profitability depends mainly on refresh cost, not acceptance. Moving the same edit later shortens the invalidated suffix and reduces speedup from 1.30× to below parity even as acceptance rises from 88% to 100%. A cost-only gate calibrated at $S^* = 16$K therefore sends long-suffix turns to StaleSpec and the rest to B1. On a heterogeneous Qwen3-30B-A3B workload, it improves always-on 1.02× to 1.13×, within 0.8% of the 1.14× cell oracle. Generation-budget sensitivity follows the same cost boundary. As G increases, gains decay toward parity, with the lower-ρ cell crossing between $G = 32$ and 64 and the higher-ρ cell between $G = 128$ and 256; beyond each crossover, the cost gate routes larger-G turns to B1 (Appendix E.4).

Why retain target self-drafting? Cheaper proposal operators save draft cost but can lose more fresh-verified work than they save. Across smaller, quantized, layer-skipped, scheduling, and trajectory-repair alternatives, none improves committed latency over its matched target-self baseline. Appendix D.3 gives the full controls.

Long-context serving with rotation-on-read. In the fixed-mode 110K-token concurrency sweep, RoR keeps KV occupancy within ±1% of B1 (Appendix E.5) and remains admissible at concurrencies where materialization does not finish. At $C = 8$, RoR achieves 3.0× throughput and 2.9× committed-turn speedup over B1 (Table 5). These closed-loop gains are capacity-enabled rather than pure per-turn algorithmic speedups: they combine per-turn refresh overlap with queue-level overlap and can therefore exceed the isolated-turn 2× bound of Theorem 1.

What determines when stale speculation helps? The results separate three effects. Semantic dependence determines how much stale-view work survives fresh verification: proposals last longer when the continuation does not yet depend on the edit. System cost determines whether that verified work translates into latency savings: a large invalidated suffix relative to the remaining decode budget creates the useful overlap window, which the admission gate detects without predicting acceptance. At long context, memory capacity becomes a third constraint, and RoR preserves that operating

Table 5: **Long-context RoR concurrency sweep.** Speedup over B1 at matched concurrency on 110K-token sessions. Materializing the stale window is competitive at low concurrency but becomes inadmissible at $C = 8$, where RoR retains $3.0\times$ throughput and $2.9\times$ committed-turn speedup. DNF denotes "did not finish".

Metric	Runtime	$C = 1$	$C = 4$	$C = 6$	$C = 8$
Throughput	materialized	1.9	4.7	2.3	DNF
	RoR	1.9	2.8	1.8	**3.0**
Committed turn	materialized	1.85	5.1	2.4	DNF
	RoR	1.6	4.3	2.9	**2.9**

region by aliasing rather than copying the stale window. Together, these results distinguish *proposal quality*, *latency profitability*, and *memory capacity* as separate factors governing temporal KV reuse.

5 Related Work

Speculative decoding and verified degraded views. Speculative sampling (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023) includes learned heads, block- or semi-autoregressive proposals, and serving-aware schedules (Li et al., 2024; 2025b; Chen et al., 2026; Cheng et al., 2026; Liu et al., 2024; Li et al., 2026; Huang et al., 2025). These methods first establish a correct edited-context view and therefore leave its refresh cost intact. TriForce, MagicDec, and Vegas self-speculate over spatially sparse fresh state (Sun et al., 2024; Sadhukhan et al., 2025; Yue et al., 2026); SpecPV degrades the verifier view within a turn (Tan et al., 2025). StaleSpec shares verifier-exact transport but addresses a different missing-state barrier. After a non-append edit, the exact suffix KV does not yet exist; StaleSpec drafts from prior-turn temporal state while that suffix is recomputed. Lossless decode-side drafters remain complementary: StaleSpec first converts refresh time into a fresh-verified prefix, after which EAGLE-3 or DFlash accelerates the residual fresh continuation (Sec. 4).

KV reuse, editable state, and agent serving. Exact prefix reuse, cache-aware routing, PD disaggregation, and agent schedulers optimize bit-exact hits and resource placement (Kwon et al., 2023; Zheng et al., 2024; Qin et al., 2025; Zhong et al., 2024; Agrawal et al., 2024; Abhyankar et al., 2024; Li et al., 2025a; Pan et al., 2025; Luo et al., 2025), but none removes the suffix refresh after a non-append edit. CacheBlend and position-independent caching partially recompute or recombine stale KV (Yao et al., 2025; Hu et al., 2025; Gim et al., 2024), trading fidelity for latency. CacheSlide targets shifted segments in agent prompts with position-aware correction (Liu et al., 2026), while Models Take Notes at Prefill studies editable and composable cached representations through errata and repositioning (Li, 2026). These approaches seek a directly usable edited cache and evaluate empirical fidelity. StaleSpec instead makes no fidelity assumption about the stale view: it is proposal-only, and every committed token comes from the exact fresh distribution.

6 Conclusion

We introduced StaleSpec, which uses stale KV after a non-append edit as a self-speculative proposal while refreshing the exact suffix in parallel. Fresh verification preserves the exact refreshed-context decoding distribution, so staleness affects only acceptance and latency.

Across controlled and deployment-shaped workloads, StaleSpec improves committed latency and throughput, composes with lossless decode-side drafters, and scales to long contexts with rotation-on-read. More broadly, invalidated model state need not be discarded or trusted approximately: when isolated behind exact verification, it can remain useful as speculative computation.

Reproducibility Statement

Section 3 states the assumptions underlying our guarantees, and Appendix F provides the complete proofs. The appendices document data provenance and trace construction, serving and measurement protocols, implementation details, exactness audits, and memory/concurrency experiments.

AI use statement

We used generative AI tools to assist with debugging existing code and correcting grammar in the manuscript. The authors reviewed all AI-assisted changes, checking code modifications for correctness and text edits for preservation of the intended meaning. We take responsibility for the final content of this work, including all text and code modified with the assistance of generative AI.

References

Reyna Abhyankar, Zijian He, Vikranth Srivatsa, Hao Zhang, and Yiying Zhang. InferCept: Efficient intercept support for augmented large language model inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 81–95. PMLR, 2024. URL https://proceedings.mlr.press/v235/abhyankar24a.html.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 117–134. USENIX Association, 2024. URL https://www.usenix.org/conference/osdi24/presentation/agrawal.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.06036.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. URL https://arxiv.org/abs/2607.05147.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/a66caa1703fe34705a4368c3014c1966-Abstract-Conference.html.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Zhang Qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25j.html.

Kaiyu Huang, Hao Wu, Zhubo Shi, Han Zou, Minchen Yu, and Qingjiang Shi. AdaSpec: Adaptive speculative decoding for fast, SLO-aware large language model serving. In *Proceedings of the 16th ACM Symposium on Cloud Computing*, 2025. doi: 10.1145/3772052.3772239. URL https://doi.org/10.1145/3772052.3772239.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Bojie Li. Models take notes at prefill: Kv cache can be editable and composable. *arXiv preprint arXiv:2606.17107*, 2026. URL https://arxiv.org/abs/2606.17107.

Hanchen Li, Runyuan He, Qiuyang Mang, Qizheng Zhang, Huanzhi Mao, Xiaokun Chen, Hangrui Zhou, Alvin Cheung, Joseph Gonzalez, and Ion Stoica. Continuum: Efficient and robust multi-turn LLM agent scheduling with KV cache time-to-live. *arXiv preprint arXiv:2511.02230*, 2025a. URL https://arxiv.org/abs/2511.02230.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, pp. 136737–136756, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Sean Lai, Xinhao Cheng, Xupeng Miao, and Zhihao Jia. AdaServe: Accelerating multi-SLO LLM serving with SLO-customized speculative decoding. In *Proceedings of the Twenty-First European Conference on Computer Systems*, 2026. doi: 10.1145/3767295.3769315. URL https://doi.org/10.1145/3767295.3769315.

Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. TurboSpec: Closed-loop speculation control system for optimizing LLM serving goodput. *arXiv preprint arXiv:2406.14066*, 2024. URL https://arxiv.org/abs/2406.14066.

Yang Liu, Yunfei Gu, Liqiang Zhang, Chentao Wu, Guangtao Xue, Jie Li, Minyi Guo, Junhao Hu, and Jie Meng. Cacheslide: Unlocking cross position-aware kv cache reuse for accelerating llm serving. In *24th USENIX Conference on File and Storage Technologies (FAST 26)*, pp. 83–99. USENIX Association, 2026. URL https://www.usenix.org/conference/fast26/presentation/liu-yang.

Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, and Ion Stoica. Autellix: An efficient serving engine for LLM agents as general programs. *arXiv preprint arXiv:2502.13965*, 2025. URL https://arxiv.org/abs/2502.13965.

Zaifeng Pan, Ajjkumar Patel, Zhengding Hu, Yipeng Shen, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. KVFlow: Efficient prefix caching for accelerating LLM-based multi-agent workflows. *arXiv preprint arXiv:2507.07400*, 2025. URL https://arxiv.org/abs/2507.07400.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation—a KVCache-centric architecture for serving LLM chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pp. 155–170. USENIX Association, 2025. ISBN 978-1-939133-45-8. URL https://www.usenix.org/conference/fast25/presentation/qin.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/13f972adf12bdf886583d48cd528002f-Abstract-Conference.html.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31, pp. 10107–10116, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/c4127b9194fe8562c64dc0f5bf2c93bc-Abstract.html.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. TriForce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=HVK6nl3i97.

Zhendong Tan, Xingjun Zhang, Chaoyi Hu, Junjie Peng, and Kun Xia. SpecPV: Improving self-speculative decoding for long-context generation via partial verification. *arXiv preprint arXiv:2512.02337*, 2025. URL https://arxiv.org/abs/2512.02337.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Yikang Yue, Yuqi Xue, and Jian Huang. Vegas: Self-speculative decoding with verification-guided sparse attention. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=vht4Pn0swG.

Z.ai. GLM-4.7-Flash. Official model card, 2026. URL https://huggingface.co/zai-org/GLM-4.7-Flash. Accessed 2026-09-05; the model card cites the GLM-4.5 technical report, arXiv:2508.06471.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210. USENIX Association, 2024. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

APPENDIX INDEX

Appendix A. Algorithm and Supplementary Material p. 13
Stale-view assembly for edited prompts.

Appendix B. Datasets, Trace Construction, and Experimental Protocol p. 13
Data provenance, acceptance statistics, controlled edits, exactness audits, and long-context serving protocols.

Appendix C. Additional Acceptance Analyses p. 18
Model scale, compaction and truncation controls, and positional re-alignment.

Appendix D. Additional Serving Analyses p. 18
Backend composition, dense-model sweeps, proposal alternatives, and end-to-end agent-turn serving.

Appendix E. Serving Implementation and Measurement Protocol p. 20
Engine configuration, turn scheduling, admission, generation budgets, rotation-on-read, and measurement procedures.

Appendix F. Proofs p. 25
Distribution exactness, latency bounds, proposal substitution, and bounded downside.

A ALGORITHM AND SUPPLEMENTARY MATERIAL

Algorithm 1 appears in the main paper. Additional analyses, complete protocols, implementation details, and operating-region studies follow below. The appendix follows the experimental logic of the main paper while keeping acceptance diagnostics separate from committed serving measurements.

A.1 STALE-VIEW ASSEMBLY FOR EDITED PROMPTS

For tokenized prompts, write the previous and edited contexts as

$$c = [P, \Delta_{\text{old}}, S],$$

and the edited prompt as

$$c' = [P, \Delta_{\text{new}}, S],$$

where P and S are the longest common prefix and suffix and Δ is the enclosed rewritten span. The assembled proposal cache is

$$K_q = [K_{\text{exact}}(P, \Delta_{\text{new}}),\ \text{Realign}(K_{\text{stale}}(S))]\,,$$

The concurrent path independently constructs the fresh cache $K_p(c')$ used for verification. The implementation conservatively encloses everything between the first and last modified token in one rewritten span. Thus, contiguous insertions, deletions, substitutions, memory or tool updates, and history compaction are handled uniformly. Disjoint unchanged regions inside that span are recomputed rather than reused; this affects proposal efficiency, not correctness.

B DATASETS, TRACE CONSTRUCTION, AND EXPERIMENTAL PROTOCOL

B.1 DATA PROVENANCE

The controlled mechanism suite contains 318 context–edit instances: 186 SWE-agent source turns, 108 LongBench multi-document edits, and 24 BFCL tool-registry updates. Held-out SWE-rebench and TerminalBench windows are disjoint from this suite and are used only for the agent-serving

Algorithm 2 Stale-view assembly for an edited prompt.

Input: previous tokens c, edited tokens c', stale cache K_{stale}
Output: assembled proposal cache K_q
compute token-level longest common prefix length k
compute token-level longest common suffix length m
$s_{\text{old}} \leftarrow |c| - m$
$s_{\text{new}} \leftarrow |c'| - m$
$\Delta_{\text{pos}} \leftarrow s_{\text{new}} - s_{\text{old}}$
reuse the KV of the common prefix $c_{<k}$ exactly
if $s_{\text{new}} > k$ **then**
    exact-prefill the edited span $c'_{k:s_{\text{new}}}$
    obtain exact KV for $c'_{<s_{\text{new}}}$
else
    retain the reused common-prefix KV
end if
copy the surviving suffix KV from $K_{\text{stale}}[s_{\text{old}} : |c|]$ to positions $[s_{\text{new}} : |c'|]$
if $\Delta_{\text{pos}} \neq 0$ **then**
    re-align copied suffix keys to their new positions
end if
concatenate the exact prefix-and-edit KV with the re-aligned stale suffix
return assembled proposal cache K_q

Table 6: Operating-region summary. Each row identifies the measured system condition, its effect on stale-view overlap, and the corresponding exact execution path.

Case	Mechanism of failure	What serves it instead
Compaction-heavy traffic	continuation often depends immediately on the rewritten summary ($\mathbb{E}[A] = 5.5$–8.1)	the cost gate selects between STALESPEC and exact B1 according to the available refresh-overlap budget
Saturated one-GPU colocation	proposal, refresh, verification, and continuation passes contend with other turns; $\varphi > \varphi_0$	exact B1 execution or a dedicated refresh pool
Chunk-relative positions (DCA)	keys stored under sawtooth rotation; no exact re-alignment	verification-absorbed fallback (Prop. 1 intact, lower α)
Long or variable-length outputs	ordinary decode occupies a larger share of turn time as G grows, reducing ρ	gate using the deployment-local expected horizon and measured refresh/decode rates

evaluation. The long-context ladder uses a separate corpus of memory-update and insertion edits. Sample counts and result-specific protocols are given in the relevant subsections below.

The common suite uses identical trace ids and deterministic shards across Qwen3-8B, Qwen3-32B, and Llama-3.1-8B-Instruct. Per model it contains 30 append, 60 elision, 48 insertion, 72 swap, 36 reorder, 24 memory-update, and 48 compaction traces. Shifted edits are additionally evaluated without positional re-alignment, giving 606 model–edit–view measurements per model. For Table 4, selected Qwen3-8B cells additionally pool an independent run from the same builder and protocol; the 32B and Llama columns remain the strictly sample-matched cross-model comparison.

B.2 Acceptance Statistics for Table 4

Per-cell N follows the edit counts in Appendix B.1; the additional Qwen3-8B pooling is disclosed there. Confidence intervals use trace-level resampling (one scalar $\widehat{\mathbb{E}[A]}$ per trace), 10,000 iterations, 95% CI (2.5/97.5 percentiles), statistic = median, and a fixed seed.

For each trace, we estimate accepted-prefix length under the actual stale-view proposal process using $R = 32$ independent autoregressive proposal trajectories,

$$x^{(r)}_{1:H_r} \sim q, \qquad r = 1, \ldots, R.$$

from the assembled stale view using the same sampling stack as the serving configuration. Each trajectory terminates at EOS or the per-trace generation cap (256 tokens for append; 384 for memory-update, swap, and reorder; and 512 for elision, insertion, and compaction).

For each proposal trajectory, the proposal pass provides $q_j(x_j^{(r)} \mid x_{<j}^{(r)})$. A causal fresh-target pass over the same proposed prefix computes $p_j(\cdot \mid x_{<j}^{(r)})$ at every position. Both distributions include the same temperature and top-p processing used by speculative transport. The conditional acceptance probability of proposal token $x_j^{(r)}$ is

$$a_j^{(r)} = \min\left\{1, \frac{p_j(x_j^{(r)} \mid x_{<j}^{(r)})}{q_j(x_j^{(r)} \mid x_{<j}^{(r)})}\right\}.$$

Let A denote the number of consecutive proposal tokens accepted before the first rejection, truncated by the realized proposal horizon. For a sampled proposal trajectory, define the prefix-survival weight

$$W_k^{(r)} = \prod_{j=1}^{k} a_j^{(r)}, \qquad k \le H_r.$$

Conditioned on the proposed tokens, $W_k^{(r)}$ is the probability that the first k proposal tokens all survive verification. Hence, accounting for the realized proposal horizon H,

$$\Pr(A \ge k) = \mathbb{E}_{x_{1:H}\sim q}\left[\mathbf{1}[k \le H] \prod_{j=1}^{k} a_j\right].$$

At each position along a proposal prefix,

$$q_j(x_j \mid x_{<j}) a_j = \min\{q_j(x_j \mid x_{<j}), p_j(x_j \mid x_{<j})\}.$$

Thus, averaging the product of acceptance probabilities over trajectories sampled autoregressively from q gives the exact speculative prefix-survival probability.

We estimate the survival curve as

$$\widehat{S}(k) = \frac{1}{R}\sum_{r=1}^{R} \mathbf{1}[k \le H_r]\, W_k^{(r)},$$

and the expected accepted-run length for each trace as

$$\widehat{\mathbb{E}[A]} = \sum_{k\ge 1} \widehat{S}(k) = \frac{1}{R}\sum_{r=1}^{R}\sum_{k=1}^{H_r} W_k^{(r)}.$$

This averages the conditional survival probabilities rather than sampling the accept/reject uniforms themselves, reducing Monte Carlo variance while preserving the exact expectation. No independence assumption across proposal positions is used.

The survival curves in Fig. 2 use $\widehat{S}(k)$ from the same proposal rollouts. Table 4 reports the median per-trace $\widehat{\mathbb{E}[A]}$ with trace-bootstrap confidence intervals. First-rejection position under greedy decoding, where reported separately, is a distinct diagnostic and is not used to estimate $\mathbb{E}[A]$.

B.3 Controlled-edit generation rules

All edits are deterministic given a per-trace seed and use real source content whenever available. Elision shortens an earlier tool observation; memory update replaces a stale observation, document segment, or tool-registry entry with a later version; insertion injects a later observation as an asynchronous result; swap replaces a size-matched passage; reorder permutes existing passages; and append uses the real next trajectory step. Compaction replaces an early dialogue span with one fixed Qwen3-8B-generated summary shared verbatim across models. This may mildly favor Qwen3-8B, but the cross-model compaction confidence intervals overlap.

Held-out serving traces come from disjoint upstream row windows, exclude all source ids used by development traces, and have zero source-id intersection with the controlled suite. Exact token budgets, edit depths, and deterministic seeds are released with the trace builder.

Table 7: **KV-packed PD serving on LongBench document-drop edits at concurrency** $C = 8$, summarized in Section 4. Throughput uplift is over the full queue-inclusive closed loop, averaged over five independent runs. Composed p95 speedup compares matched concurrent turn-timing diagnostics; First-proposal speedup measures pre-verification availability relative to B1. The composed latency includes fresh verification and continuation.

Model	Throughput uplift	Composed p95 speedup	First-proposal speedup
Qwen3-30B-A3B	$+19\%$	$1.85\times$	$20\times$
Qwen3-235B-A22B	$+19\%$	$1.44\times$	$22\times$
GLM-4.7-Flash	$+11\%$	$1.36\times$	$19\times$

B.4 Transport-correctness audit (TV) protocol

Sampling defaults for all stochastic-decoding evaluations are temperature $T = 0.9$ and top-$p = 0.95$, unless otherwise specified. We audit Qwen3-8B on the first **16 traces** of the mechanism suite, consisting of real SWE-agent elision turns from the 8K bucket. The audit is conducted in fp32. For each trace, we generate an on-policy proposal of up to 64 tokens from the assembled stale distribution and audit the first **32 positions**, yielding approximately 507 audited positions.

The purpose of this experiment is to verify that the stochastic transport implementation, including full-distribution residual sampling after temperature and top-p processing, reproduces direct sampling from the fresh target distribution. It is therefore an implementation-correctness audit rather than a serving-latency measurement. At each audited position, we materialize the full post-processor target and proposal distributions, p_i and q_i, over the vocabulary. For a proposal token $x_i \sim q_i$, the acceptance test uses

$$\min\left\{1, \frac{p_i(x_i)}{q_i(x_i)}\right\}.$$

At the first rejection, the replacement token is sampled from the normalized positive residual

$$r_i(y) = \frac{(p_i(y) - q_i(y))_+}{\sum_z (p_i(z) - q_i(z))_+},$$

which requires the full q_i.

At each position, we draw **20,000** emissions through this transport rule and, independently, 20,000 direct samples from the processed fresh target p_i. Both proposal and target distributions use the same temperature and top-p processing. We compare the resulting empirical histograms using

$$\widehat{\mathrm{TV}}_i = \frac{1}{2}\sum_v \left|\hat{p}_{\mathrm{emit},i}(v) - \hat{p}_{\mathrm{direct},i}(v)\right|,$$

whose resolution is $0.5/20{,}000$. The audit uses one fixed seed for reproducibility. Because the reference is itself a 20K-sample Monte Carlo histogram, an exact implementation is expected to exhibit small nonzero TV from finite-sample variation. The measured mean TV is 0.0009.

We additionally test the audit's detection power with an injected *logit-processor stack mismatch*: the acceptance ratio and residual are computed after temperature scaling only, whereas the served target additionally applies top-p truncation. This increases mean TV to 0.0057, approximately $6\times$ the matched-stack value. A synthetic self-test further requires

$$\mathrm{TV}_{\mathrm{ok}} < 0.015, \qquad \mathrm{TV}_{\mathrm{bug}} > 3\,\mathrm{TV}_{\mathrm{ok}}.$$

B.5 KV-packed PD-serving protocol

Data and edit. The density workload uses public LongBench multi-document QA contexts from HotpotQA, 2WikiMQA, and MultiFieldQA-en, with up to 24 passages. After each model's chat template, selected prompts are approximately 15–16K tokens. We deterministically remove the *second* passage. This is a single-document-drop edit at an early position (roughly 5% context depth) and invalidates a long median suffix of approximately 12K tokens, modeling a retrieval-set update in which an early retrieved document is removed before the next turn. Every model sees the same raw

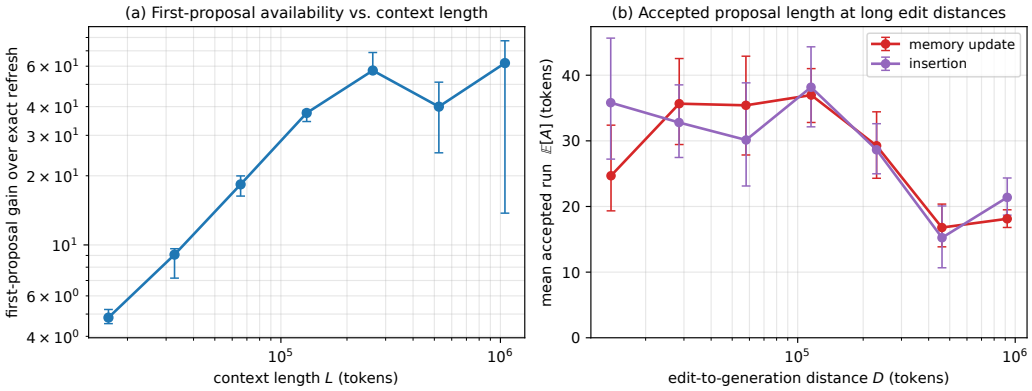

Figure 3: **Long-context scaling on YaRN-extended Qwen3-8B (bf16).** We use an offline serial harness and place each edit at position $0.1L$, so both the invalidated suffix and the edit-to-generation distance grow with context length. Generation uses the natural per-trace horizon (median 54, IQR 26–103, capped at 512 tokens). (a) First-proposal availability is T_p/T_{prop}, where T_p is the measured B1-matched suffix-refresh time and T_{prop} includes stale-view assembly and the first draft step. Points are ratios of bucket medians over $N = 40$ traces for 16K–256K and $N = 19$–20 for 512K–1M. (b) Mean accepted run $\mathbb{E}[A]$ is plotted against the measured edit-to-generation token distance D for memory-update and insertion edits ($N = 9$–20 traces per point and edit type). Error bars are 95% bootstrap confidence intervals from 2,000 resamples. First-proposal availability is measured before verification; committed latency is reported separately in the main controlled-latency results and Table 7.

examples and the same passage-level edit; tokenized lengths vary slightly because model tokenizers differ. The fixed passage index controls edit depth and refresh work across paired methods and models. A separate fixed-context depth sweep in Appendix E.3 measures how the invalidated suffix changes the profitability boundary. Throughput, p95, and first-proposal results are reported in Table 7.

Serving protocol. Generation uses $G = 64$ greedy tokens. Resident-donor counts are $M = 43/32/81$ for Qwen3-30B-A3B, Qwen3-235B-A22B, and GLM-4.7-Flash, respectively. Occupancy is used KV blocks divided by total GPU blocks after donor initialization. The closed loop holds $C = 8$ turns in flight and measures $N = 48/24/48$ completed turns per run, respectively. Each cell is repeated across five independent closed-loop runs. Throughput is N divided by full-loop wall time, and p50/p95 are computed within each run and then averaged across the five runs. The prefill pool is a serial FIFO with one refresh in flight and no refresh batching, a conservative choice shared by B1-disagg and StaleSpec.

Timing and correctness. Both pools execute concurrently. Suffix-KV transfer is charged using the measured 780 GB/s NVLink bandwidth. The table therefore reports a deployment-shaped concurrent timing composition, not a monolithic cross-process live-KV-handoff measurement. Correctness is gated separately by fp32 committed-versus-fresh parity and the stochastic transport audit in Appendix B.4.

B.6 Long-context ladder

Model and RoPE configuration. Qwen3-8B and Qwen3-32B are evaluated in bf16 (fp32 is memory-infeasible at these L). Native `rope_theta` 10^6 is unchanged. YaRN is a static config override {`rope_type: yarn`, `original_max_position_embeddings: 32,768`} with per-bucket factor: none (16K/32K, native 40,960 window), 2 (64K), 4 (128K), 8 (256K), 16 (512K), 32 (1M) — a static frequency rescale, so the $R(\Delta)$ re-rotation remains exact. GLM = THUDM/glm-4-9b-chat-1m-hf run *natively* (no extrapolation; `rope_theta` 10^4, 1M window, interleaved partial rope with rotary dim 64), gated by an fp32 re-rotation exactness unit test before measurement. For the accepted-run analysis in Fig. 3(b), we use the stale-view proposal estimator of Appendix B.2 with $R = 32$, temperature 0.9, top-$p = 0.95$, a maximum proposal horizon of 512 tokens, and EOS enabled. Traces with realized horizons shorter than 8 tokens are omitted. **GLM 512K anchor**: N=17

Table 8: **Controlled committed speedup on dense GQA models.** Each entry reports $T_{\mathrm{B1}}/T_{\textsc{StaleSpec}}$ for $\rho = 0.3/1/3$ under matched model, edit, context, hardware, and generation settings. Deployment-scale MoE and MLA results are reported separately in Table 7.

Model	Context	Elision	Swap
Qwen3-8B	8K	1.04/1.08/1.08	1.03/1.05/1.05
Qwen3-32B	8K	1.04/1.09/1.09	1.04/1.09/1.10
Qwen3-32B	32K	1.02/1.08/1.13	1.00/1.02/1.03

traces (9 memory-update and 8 insertion) completed within the job wall-time; completed traces define the reported anchor. Median T_p is 1105.9 s versus 5.49 s median time-to-first-proposal, yielding $201.3\times$ availability; a trace-level bootstrap (2,000 resamples) gives a 95% CI of $[129, 230]$.

C Additional Acceptance Analyses

Scale. Coarse mean acceptance is similar across model scales, but accepted runs on low-dependence edit classes are sometimes 1.5–$2\times$ shorter at 32B, consistent with a sharper target distribution.

Compaction and truncation control. The expanded compaction set gives $\mathbb{E}[A] = 5.5$ $[4.2, 7.4]$ for Qwen3-8B and 5.9 $[4.4, 7.2]$ for Qwen3-32B; the confidence intervals overlap, and only about 30% of turns reach $\mathbb{E}[A] \geq 10$. All Table 4 values come from the untruncated offline protocol. Values from fixed paired subsets used by auxiliary baselines must not replace these canonical aggregates.

Positional re-alignment. Re-alignment improves acceptance by up to 0.80 on compaction and is nearly neutral on pure reorderings. In the serving prototype, disabling re-alignment can change a fully accepted window into immediate rejection. The architecture boundary is summarized in Table 6.

D Additional Serving Analyses

D.1 Composition protocol and matched-backend accounting

Table 2 reports the full $C{\in}\{8, 16\}$ grid summarized in Section 4. We use Qwen3-8B with a 1:1 PD-disaggregated deployment and hold the model, trace ids, request order, hardware allocation, concurrency, and generation settings fixed within every pair. The three decode backends are ordinary decoding, EAGLE-3, and DFlash.

Each entry is the committed-p50 latency reduction obtained by adding StaleSpec to the named backend: B1 versus StaleSpec, B1+EAGLE-3 versus StaleSpec+EAGLE-3, and B1+DFlash versus StaleSpec+DFlash. Consequently, the standalone acceleration of EAGLE-3 or DFlash is excluded from our reported uplift. The outer stale-view proposal is transported to the same fresh target as in the base method, so composition remains distribution-exact in every evaluated cell, including DFlash/SWE-smith at $C{=}16$, which yields a 14% incremental committed-p50 reduction.

The composition is complementary rather than redundant. In every composed path, target self-drafting remains the stale-side proposal source while the edited suffix is refreshed. Fresh verification then commits M tokens and produces the target state required by EAGLE-3 or DFlash; the auxiliary backend accelerates only the residual $G - M$ fresh continuation. All reported values therefore compare complete matched serving paths rather than adding independent microbenchmark speedups.

D.2 Controlled dense-model sweep

Table 8 expands the controlled committed-latency result from Section 4 across dense GQA models, edit types, context lengths, and refresh-to-generation cost ratios.

End-to-end stochastic serving. On Qwen3-30B-A3B with temperature 0.9, top-$p = 0.95$, PD 1:1, and $C = 8$, the timed path retains the post-processor proposal distributions, samples the positive residual on rejection, verifies, and continues. Across the measured agent-context cells it improves committed latency by 8–11% relative to B1; proposal-distribution capture is 1.5% of turn latency. For

Table 9: **Resource-factorized stale-proposal controls.** Inner acceptance measures imitation of the target stale view, while outer acceptance is measured against the unchanged fresh target. Each row uses its own matched subset and target-self reference; cross-row absolute values are not compared.

Proposal path	Resource changed	Matched result	Interpretation
Persistent dual-cache drafter	capacity	$A_{\text{in}} = 6.56$; $A_{\text{out}} = 4.56$ vs. self 9.88	State delivery is efficient (94.4% context recovery, +2.78% HBM), but the small proposal model remains capacity-limited.
Full-width W4 twin	precision	$A_{\text{in}} = 18.91$ vs. 19.17; $A_{\text{out}}/A_{\text{self}} = 0.84$	Full capacity recovers the stale teacher, but numerical perturbation reduces fresh overlap; the tested W4 stack is also slower (48.2 vs. 36.3 ms/token).
Layer-skipped target	depth	best $A_{\text{out}} = 1.12$ vs. self 15.43	Acceptance remains at 0.07–0.11× after calibrated and oracle activation-scale correction, isolating missing transition computation.
Elastic scheduling	critical-path charge	no measurable gain at 8K or 16K	Continuous batching already hides almost all stale self-draft latency under the compute-bound refresh.
Tail/tree/seed repair	trajectory repair	no committed gain	Oracle support exists, but realized tails follow the wrong trajectory or cost at least the sequential work they replace.

GLM, the main serving table uses the fused greedy verifier, while the stochastic transport is validated separately by the exactness audit.

Fresh small-drafter baseline. The natural rival drafter is a small model reading the *fresh* edited context. On a fixed paired subset with on-policy continuations, it is weak on both acceptance and time to first proposal. Acceptance is *model-gap-bounded*: $\mathbb{E}[A] \leq 3.3$ in every edit class (0.3–3.3 vs. 5.5–56.5 stale-self; the drafter wins on $\leq 12\%$ of traces in any class and 0% on insert/elide/memory-update; under greedy decoding its first mismatch arrives at position ~ 1). And its first proposal still pays an $O(L)$ prefill: measured drafter prefill $0.49/1.27/1.84$ s at $8/16/32$K and growing, versus our $O(|\Delta|)$ delta patch (0.13–0.15 s at 32K on the serving path, L-independent). The cross-class spread is diagnostic: stale-self spans $10.7\times$ across classes (edit-dependence-bounded) while the fresh drafter spans $12.9\times$ around a floor of $\sim$1–3 tokens (model-gap-bounded)—complementary failure modes, and only ours co-locates with the edits that matter. Distilled decode-side drafters (EAGLE-3 (Li et al., 2025b) and DFlash (Chen et al., 2026)) are instead complementary after fresh verification: they accelerate the residual fresh continuation in Table 2, rather than replacing target self-drafting from the stale view.

D.3 Resource-factorized proposal alternatives

We evaluate whether target self-drafting can be replaced along independent resource axes while keeping the fresh verifier unchanged. The experiments use different matched subsets, so Table 9 reports every alternative together with its own target-self reference. We compare within rows only and do not treat unmatched absolute values as a global ranking.

Table 9 supports a state–operator interpretation of the stale proposal: the cache supplies the state, whereas the target model supplies the transition operator that produces a long autoregressive trajectory. Persistent dual-cache reuse substantially closes the context-interface gap, but the evaluated reductions in model capacity, numerical precision, or depth shorten the fresh-target accepted trajectory. The proposal-replacement criterion (Corollary 1) explains why lower raw proposal cost is insufficient when the sacrificed verified work was already produced off the critical path. Across these tested controls, no alternative provides a better measured replacement for target self-drafting. Because the experiments use different matched subsets and the search is not exhaustive, this is a design conclusion for the evaluated alternatives rather than a claim of global optimality.

We also mined multi-version rollback lineages in SWE-agent and OpenHands. Candidate historical donors occur in 52–55% of edited turns, but fuzzy donor selection improves modeled committed latency by only approximately 1–2%. Most larger gains arise from exact content reappearance and reduce to ordinary content-addressed KV reuse. We therefore leave lineage-aware donor selection to workloads with larger semantic reversions.

D.4 End-to-end agent-turn serving: setup and CacheBlend HKVD details

This subsection details Table 3.

Setup. We replay held-out edited agent contexts as end-to-end model-serving requests on PD-disaggregated 1:1 pools, with one prefill and one decode engine per model (Qwen3-235B-A22B uses TP4 per pool). Committed latency spans edit arrival to the G-th verified token and includes exact suffix refresh, stale-view assembly, proposal decoding, fresh verification, continuation, scheduling, and the charged P→D transfer term. Replay fixes the context and request order for paired comparison. External tool execution is excluded, and generated turns are not fed into later contexts; the evaluation is therefore an end-to-end model-serving benchmark rather than a closed-loop agent-task benchmark. The benchmark uses $C{=}8$ turns in flight, greedy decoding, $G{=}64$ new tokens per turn, and $N{=}32$ turns per cell. Every method serves the same turns in the same request order within a run, on the same hardware and scheduler. SWE-smith / SWE-rebench / TerminalBench families come from real OpenHands and terminal-agent trajectories with the edit synthesized by the supersede construction. Every cell is the mean of three independent runs. In Table 3, $\pm$ is omitted when the run-level std rounds to zero at the displayed precision; nonzero std values are shown explicitly.

Baselines and CacheBlend HKVD. B1 re-prefills exactly the invalidated suffix on the prefill pool (longest-common-prefix reuse; P→D handoff charged with an NVLink transfer-time model). **PromptCache** reuses the stale KV verbatim with no refresh and no verification. **CacheBlend** implements the actual HKVD policy: per-token stale-vs-fresh KV deviation at layer 1 ranks the reused span (at the estimation layer the recomputed KV equal the fresh ones, so this is the policy's exact selection signal), the top $r{=}15\%$ positions are spliced with fresh KV, and the turn pays the policy's real costs—the in-path selection (8–21 ms/turn), a real prefill job of $(r{+}1/L)\cdot S$ tokens on the prefill pool (the FLOP match of the layer-1 estimation pass plus the top-r selective recompute), and the splice itself. Spliced values are the fully fresh KV, an upper bound on the blended recompute. StaleSpec drafts from the stale KV while the refresh runs, then rejection-verifies every token against the refreshed context.

Why a 15% recompute does not preserve quality. Two observations defuse the natural objection. (i) A fixed-leading-span variant (same r, no selection) is indistinguishable from HKVD selection in both speed and fidelity even though the two policies select nearly disjoint positions (overlap 8–16%, the random-overlap level for $r{=}15\%$): under position-shifting edits, per-token KV deviation is not sparse, so no 15% patch set can certify the output. (ii) The blended cache's *per-step* distortion is indeed small (oracle-ranked offline sweep: 1.6–2.9% teacher-forced disagreement), consistent with CacheBlend's quality claims in its chunk-reuse setting, but it is uncertified—a free-running agent turn diverges at the first uncorrected flip, which is what the trajectory deviation in Table 3 measures. The fixed-span variant additionally loses the committed advantage entirely on position-shifting edits (reorder: $0.60\times$), and CacheBlend shows heavy splice tails at scale (p95 11.2–11.4 s vs. B1 3.4–4.0 s on 235B).

E Serving Implementation and Measurement Protocol

E.1 Engine configuration

- Hardware is one B200 per colocated cell. The disaggregated component measurement uses two B200s connected by NVLink at a measured 780 GB/s.
- Timing uses upstream vLLM 0.24.0; the same driver is checked on 0.18.0. The engine is in process, while tensor-parallel workers remain standard vLLM workers. Grid cells use $\mathrm{MBT} = 512$ and closed-loop concurrency uses 4096, with block size 16 and bf16 weights/KV. CUDA graphs are enabled for timing; fp32 losslessness gates are repeated under both eager and graph modes. Chunked prefill is not graph-captured.
- Automatic prefix caching is disabled for the choreography to avoid hash pollution of borrowed blocks. B1 receives identical prefix reuse explicitly by seeding the resident prefix, so it is refresh-matched rather than APC-handicapped.

E.2 TURN CHOREOGRAPHY AND OPTIMIZATIONS

One edited turn executes as: (i) a delta patch prefills the new edited span against the shared exact prefix; (ii) stale-view assembly reuses the prefix and vectorizes gather, relative-RoPE rotation, and scatter over the surviving suffix; (iii) proposal and exact refresh are co-batched by the engine; and (iv) one seeded verification prefill checks all proposal positions. On accept-all, that same request continues the committed stream in place. On a rejection, a corrected-token continuation finishes the turn.

Relative to the initial per-phase-request prototype, the optimized stack adds:

- **Proposal priority and budget.** Proposal work is admitted at low priority and capped at $2\lceil L_{\text{inv}}/\text{MBT}\rceil + 8$ tokens, twice the predicted refresh-window scale plus a small guard. The factor-two window and $+8$ guard are conservative caps rather than tuned hyperparameters; drafting normally terminates at refresh completion before the cap binds.
- **Verify-and-continue fusion.** The common accept-all path removes a second request, admission, KV re-seed, and boundary-block copy.
- **Vectorized re-alignment.** GPU ranges replace per-token host loops; the relative rotation table is computed once per turn. At 32K this reduces t_δ from 54 to 14 ms.
- **CUDA graphs and block-lifecycle hardening.** Short decode phases use graphs; explicit losslessness and forced-proposal acceptance gates catch both committed-output errors and reject-all proposal-path failures.
- **TP-transparent block operations.** Block-table surgery remains in the in-process scheduler. At TP>1, physical KV operations dispatch to all shards through `collective_rpc`; the identical code path is tested at TP 1–8 for GQA and MLA layouts.

The following sections report edit-depth sensitivity and the orthogonal rotation-on-read (RoR) runtime.

E.3 BUDGET-AWARE ADMISSION AND EDIT-DEPTH SENSITIVITY

Policy and implementation. At admission, the longest-common-prefix exposes the invalidated suffix S. The runtime maintains exponential moving averages of suffix-prefill throughput and per-token decode latency and forms $\widehat{\rho} = \widehat{T}_p/(G\widehat{t}_d)$. A request uses STALESPEC only when $\widehat{\rho} \geq \tau_\rho$ (equivalently $S \geq S^*$ within a fixed deployment); otherwise it follows B1. The initial decision uses cost signals only and never predicts acceptance. Thresholds are keyed by edit signature. An optional causal outcome-regret update adjusts τ_ρ multiplicatively, clipped to $[0.5, 8]$, with ϵ-exploration for audit coverage. Every decision is logged, and `STALESPEC_GATE=0` recovers always-on evaluation behavior. The gate performs $O(1)$ host arithmetic plus one audit record and is below the resolution of the turn-level measurements.

Fixed-context depth calibration. To isolate the refresh budget, we re-chunk one real agent context to approximately 25.8K tokens and splice the same approximately 400-token edit at five depths. The document, total context length, $G = 64$, $C = 8$, PD 1:1 hardware, greedy decoding, and $N = 32$ turns remain fixed; only the invalidated suffix changes. Qwen3-30B-A3B is the lightest-refresh deployment in the suite. T_p is the measured B1 suffix-refresh critical path, and acceptance is the fraction of drafted tokens committed unchanged by rejection verification (Table 10). Committed gain falls from $1.30\times$ at 5% depth to $0.59\times$ at 95% depth while acceptance rises from 88% to 100%, so the parity crossover—not realized acceptance—defines the admission boundary used by the online gate.

Mixed-workload dispatch. The default $S^* = 16$K is fixed from the depth calibration rather than selected from the mixed-workload outcomes.

Budget-gated execution captures nearly all of the available speedup on this heterogeneous mixture. The result shows that the refresh-to-decode cost ratio is sufficient to select the faster exact path without observing proposal acceptance or post-hoc cell latency.

Table 10: **Fixed-context edit-depth sweep on Qwen3-30B-A3B at $C = 8$.** Moving the same edit later shortens the invalidated suffix and lowers refresh cost, so the committed gain decays and eventually falls below parity—but *these below-parity cells are never served by* STALESPEC. The gate threshold S^*=16K coincides with the parity crossover: it routes the two profitable early-position cells (long suffix) to STALESPEC (SS) and the three sub-parity late-position cells (short suffix) to B1 (last column). Acceptance meanwhile rises from 88% to 100%, so an acceptance-based dispatcher would invert this decision; the cost-only $\widehat{\rho}$ gate does not. The below-parity numbers are therefore the calibration curve for the gate, not an operating point of the method.

Edit depth	Invalidated suffix	T_p (s)	B1 p50	committed ×B1	first ×B1	accept	gate→
5%	24,759	2.64	3.37s	1.30×	11.6×	88%	SS
25%	20,295	1.72	2.58s	1.19×	7.7×	97%	SS
50%	14,832	0.53	1.62s	0.87×	2.4×	100%	B1
75%	8,285	0.26	1.36s	0.70×	1.2×	100%	B1
95%	1,872	0.20	1.12s	0.59×	0.7×	100%	B1

Table 11: Cost-aware admission on the mixed Qwen3-30B-A3B workload (PD 1:1, $C = 8$, $G = 64$; five-run aggregate). The gate chooses between two exact paths using pre-execution cost signals and reaches within 0.8% of the cell-level oracle.

Policy	Speedup over B1	Relative regret
B1	1.00×	–
Always-on STALESPEC	1.02×	–
Budget-gated	**1.13×**	**0.8%**
Cell-level oracle	1.14×	0%

E.4 GENERATION-BUDGET DISTRIBUTION AND SWEEP

Real agent turns are short. The fixed budget $G = 64$ used in trace replay is not a floor on realistic output length but sits above the typical turn. Across 9,858 assistant turns in 400 real SWE-agent trajectories, tokenized with the serving tokenizer, the per-turn output length has median 19 tokens, mean 42, and 90th percentile 102; 82% of turns emit ≤ 64 tokens (93% ≤ 128, 98% ≤ 256). Agent turns are dominated by short tool calls and terse reasoning rather than long generations, so a shorter output raises $\rho = T_p/(G\, t_d)$ for a fixed refresh workload and therefore expands the profitable region; $G = 64$ is a conservative anchor—the median turn has *higher* ρ and larger predicted gain. Profitability still depends jointly on the invalidated-suffix cost and the predicted generation horizon, not on output length alone. Long-output turns (≥ 256 tokens, $\sim$2%) have lower ρ and are more likely to be routed to B1 when their configured or predicted horizon reflects the realized length.

Generation-budget sweep. Varying only G at fixed context, edit, and hardware traces the same ρ-controlled boundary along the generation budget that Table 10 traces along edit depth. G-tolerance rises with ρ: the low-ρ fast-MoE cell crosses parity between G=32 and 64, while the higher-ρ decode-bound cell tolerates larger budgets and crosses only between 128 and 256. Both peaks stay within the $S_e \leq 2$ envelope of Theorem 1, and gains eventually decay toward parity as G grows; the ordering of the two crossover points—the lower-ρ cell crossing earlier—matches the first-order dependence on $\rho = T_p/(G\, t_d)$. Past each crossover the gate routes larger-G turns to B1, and because typical turns are short (median 19 tokens, below both crossovers) the common budget sits in each regime's profitable range.

E.5 ROTATION-ON-READ DUAL-VIEW RUNTIME

Dual view and page mapping. After an edit, source token n in the reusable old suffix must be read at destination position $n + \Delta$. The stock path materializes the window into fresh pages and re-rotates every key. RoR instead registers, per request, $(\Delta, \mathrm{win}_{lo}, \mathrm{win}_{hi})$, the read-only donor block table, and $\cos$ / $\sin(\Delta\omega_d)$ constants. For block size 16, destination position p resolves to

$$\mathrm{page}(p) = \begin{cases} \mathrm{donor}[(p-\Delta)/16], & \mathrm{win}_{lo} \leq p < \mathrm{win}_{hi}, \\ \mathrm{main}[p/16], & \text{otherwise}, \end{cases}$$

Table 12: Measured decode-pool KV footprint versus concurrency. One block is 16 tokens and the B200 pool contains 63,337 blocks. RoR tracks B1 within $\pm 1\%$ because the stale window is aliased rather than copied.

C	B1	stock materialized	RoR	note
110K-token sessions				
1	13.8K	20.7K	**13.8K**	stock $+50\%$
2	13.7K	27.4K	**13.8K**	$+6.8$K/turn
4	13.6K	41.0K	**13.7K**	stock 65% pool
6	13.8K	54.9K	**13.9K**	stock 87% pool
8	13.7K	DNF	**13.8K**	only RoR/B1 admit
12	13.5K	–	**13.6K**	RoR $3.0\times$ throughput
30K TerminalBench-derived sessions				
8	34.7K	42.5K	**35.2K**	
16	34.9K	50.2K	**35.1K**	
32	34.7K	DNF	**35.2K**	RoR 2.36 turns/s
64	34.0K	–	**35.0K**	RoR 2.96 turns/s

with donor slot $(p - \Delta) \bmod 16$. Values need no correction. Window keys are scored with $q_{\text{win}} = R_{-\Delta} q$ using $(R_m q)^\top (R_\Delta R_n k) = (R_{m-\Delta} q)^\top (R_n k)$; all other columns use the ordinary query. Online softmax and verification are unchanged.

Assembly and reported modes. Each turn materializes only one boundary block containing the fresh head and unaliasable sub-block sliver, generation pages, and metadata. The shared prefix and shifted window remain refcounted aliases of resident pages. The resulting per-turn allocation is 4–33 blocks at 30K and approximately 45 at 110K, versus one full copied window per materialized turn. All RoR measurements use a fixed materialized mode or a fixed RoR mode; no admission gate or dispatch policy is applied in any reported cell.

Correctness and implementation scope. The verifier never reads the dual view; committed tokens are rejection-verified against freshly refreshed KV. RoR therefore inherits the lossless transport guarantee even if draft-side floating-point differences change acceptance. A direct rotation unit test has maximum error 4.9×10^{-4}. Two of three engine-probe families admit a direct materialized-versus-RoR equivalence test and both pass $48/48$ first-token checks with decode parity; the third contains a content edit for which both proposals are stale-limited rather than algebraically identical. The kernels are approximately 120-line patches to vLLM's Triton unified-attention 2D and split-softmax 3D paths and to the MLA decode kernel. Only MLA's 64 positional k_{pe} dimensions are transformed; the position-free latent is aliased. The RoR serving sweep uses Qwen3-30B-A3B, while MLA and TP implementations are covered by architecture-specific correctness gates. The portable materialized runtime supplies the cross-model deployment comparison, and RoR isolates the long-context memory benefit of aliasing the shifted window rather than copying it.

Fixed-mode long-context concurrency sweep. The sweep in Table 5 uses Qwen3-30B-A3B with a 131K YaRN context window and 110K-token sessions derived from real agent traces by placing the edit near the context beginning and appending identical history. Serving is PD 1:1 on two B200s, greedy, with $G = 64$ and CUDA graphs enabled. Each table entry is one closed-loop cell. The runtime summary preserves absolute B1 anchors and speed ratios. B1 committed seconds / first-token seconds / turns per second are $6.96/5.03/0.141$ at $C = 1$, $38.8/36.8/0.111$ at $C = 4$, $28.4/26.5/0.199$ at $C = 6$, and $61.4/59.4/0.120$ at $C = 8$. The B1 anchors are non-monotonic in C because each entry is a single closed-loop run whose queueing and admission realization varies; all reported speedups compare methods at matched C on the identical workload rather than across C.

Table 13: Measured total peak HBM of the portable materialized path. The premium is a conservative upper bound: the emulation keeps a fresh donor resident, whereas a physical deployment replaces it with transferred KV and retains only the stale donor plus active working state.

Model	B1 peak	StaleSpec peak	premium
GLM-4.7-Flash (MLA)	6.67 GiB	7.73 GiB	$+16\%$
Qwen3-30B-A3B	51.4 GiB	62.9 GiB	$+22\%$
Qwen3-8B (dense)	74.9 GiB	102.8 GiB	$+37\%$

At a separate 131K operating point, RoR produces its first proposal in $0.079\,\mathrm{s}$ versus $0.181\,\mathrm{s}$ for the materialized path and yields a $64\times$ pre-verification availability diagnostic over B1. Because no downstream stage is timed, this value is not presented as a committed or end-to-end speedup. Committed and KV-footprint results appear in Tables 5 and 12.

E.6 Measurement protocol

Committed E2E is wall-clock time from edit arrival to the G-th verified token. The donor cache is resident. B0 cold-prefills the full edited context; B1 seeds the same resident prefix, refreshes the invalidated suffix exactly, and then decodes. First-proposal latency is pre-verification availability and is never reported as committed TTFT. Grid cells contain 20 offered turns per (edit,L,ρ) configuration. Serving cells use the counts shown below. Every code change touching the choreography reruns: (i) fp32 committed-versus- fresh parity on three traces, $48/48$ first tokens each, under eager and graph modes; (ii) a synthetic mid-elision acceptance gate that forces the proposal path, because a reject-all bug can remain lossless while providing no acceleration; and, for RoR, (iii) the query-rotation unit and direct materialized-view engine probes described in Appendix E.5.

Instrumentation audit. Requesting `prompt_logprobs` was found to invalidate the intended seeded-KV path and trigger hidden prompt recomputation: the charge is 141 ms at 30K but approximately 2.0 s at 101K. Isolated probes attribute only 1.5 ms to block addition and 0.2 ms to KV seeding. Timed serving runs therefore disable `prompt_logprobs`; verification statistics are collected in a separate non-timed probe. Runs containing this hidden recomputation are not used for the reported latency tables.

F PROOFS

F.1 PROOF OF PROPOSITION 1

Fix a proposal prefix $x_{<i}$ and condition on it. The proposal token x_i is sampled from $q_i := q(\cdot \mid c', x_{<i})$, and verification computes $p_i := p(\cdot \mid c', x_{<i})$ under K_{fresh}. The acceptance ratio is evaluated only for the sampled token; because $x_i \sim q_i$, $q_i(x_i) > 0$ almost surely. For any vocabulary item x, the probability mass emitted through acceptance is

$$q_i(x) \min\left\{1, \frac{p_i(x)}{q_i(x)}\right\} = \min\{p_i(x), q_i(x)\},$$

with the left-hand expression needed only when $q_i(x) > 0$. Let

$$r_i := 1 - \sum_y \min\{p_i(y), q_i(y)\} = \sum_y (p_i(y) - q_i(y))_+.$$

If $r_i > 0$, define the residual distribution

$$R_i(x) := \frac{(p_i(x) - q_i(x))_+}{r_i}.$$

The total probability of emitting x is therefore

$$\min\{p_i(x), q_i(x)\} + r_i R_i(x) = \min\{p_i(x), q_i(x)\} + (p_i(x) - q_i(x))_+ = p_i(x).$$

If $r_i = 0$, then $p_i = q_i$, rejection has probability zero, and the unused residual distribution may be defined arbitrarily. Hence the emitted token has exact conditional distribution p_i for every conditioned prefix. Induction over positions gives the exact joint fresh-decoding distribution. The verification pass itself produces fresh KV for accepted tokens; after a residual or bonus token, all later tokens are decoded under K_{fresh}. □

I.i.d. acceptance intuition. If proposal positions are independently accepted with common probability α, then for a cap n and output length G,

$$\mathbb{E}[M] = \sum_{j=0}^{\min\{n, G-1\}} \alpha^j.$$

The main analysis does not assume independence and uses measured $\mathbb{E}[M]$.

F.2 PROOF OF THEOREM 1

For a realization committing M tokens, assembly and refresh must complete before verification. Verification costs t_v, after which $G - M$ tokens remain. Therefore

$$T_{\textsc{StaleSpec}}(M) \le \max\{t_\delta, \varphi T_p\} + t_v + t_d(G - M).$$

Taking expectations gives the stated bound. For a nonempty proposal window, $\varphi T_p > t_\delta$. Under the first-order model,

$$\widetilde{T}_{\textsc{StaleSpec}} = \varphi T_p + G t_d - \mathbb{E}[M] t_d = G t_d\big(1 + \rho(\varphi - \beta)\big).$$

The blocking and serving-matched exact baselines are $T_{\mathrm{B0}} = G t_d(1 + \rho)$ and $T_{\mathrm{B1}} = G t_d(1 + \varphi_0 \rho)$, yielding the displayed ratios. Retaining verification but assuming assembly is hidden, STALESPEC is faster than B1 iff

$$\varphi T_p + t_v - \beta T_p < \varphi_0 T_p,$$

which is equivalent to $\beta > \varphi - \varphi_0 + t_v / T_p$. At $\varphi_0 = \varphi = 1$, lower-order-free window capacity gives $\beta \le \min\{1, 1/\rho\}$ and hence

$$\widetilde{S} \le \frac{1 + \rho}{1 + \rho(1 - \min\{1, 1/\rho\})} = 1 + \min\{\rho, 1/\rho\} \le 2.$$

□

F.3 Proof of Corollary 1

Under matched refresh, verification, horizon, and residual-decoding terms, write the first-order expected latency of proposal path m as

$$\mathbb{E}[T_m] = T_{\text{fixed}} + \mathbb{E}[C_m^{\text{crit}}] + t_d\big(G - \mathbb{E}[M_m]\big).$$

Subtracting the expressions for a and b gives

$$\mathbb{E}[T_b] - \mathbb{E}[T_a] = \mathbb{E}[C_b^{\text{crit}}] - \mathbb{E}[C_a^{\text{crit}}] - t_d\big(\mathbb{E}[M_b] - \mathbb{E}[M_a]\big).$$

The difference is negative exactly under the inequality stated in Corollary 1. □

F.4 Bounded-downside corollary

Corollary 2 (Bounded downside)**.** *For every acceptance realization, including total rejection,*

$$T_{\text{StaleSpec}}(M) \le T_{\text{B0}} + t_v + t_\delta + (\varphi - 1)_+ T_p.$$

Thus the per-turn downside is limited to assembly and verification plus positive refresh slowdown. Under saturated colocation, B1 is the appropriate latency path; both paths remain exact.

Proof. For every realization, $M \ge 0$, so the pointwise bound in Theorem 1 gives

$$T_{\text{StaleSpec}}(M) \le \max\{t_\delta, \varphi T_p\} + t_v + G t_d.$$

Using $\max\{t_\delta, \varphi T_p\} \le T_p + t_\delta + (\varphi - 1)_+ T_p$ and $T_{\text{B0}} = T_p + G t_d$ yields

$$T_{\text{StaleSpec}}(M) \le T_{\text{B0}} + t_v + t_\delta + (\varphi - 1)_+ T_p.$$

No acceptance or verification-rate assumption is required. □