SPECULA: LOSSLESS KV-CACHE RECOVERY VIA PRECISION-DUAL DRAFTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Disaggregated LLM serving separates prefill and decode across nodes, exposing generation to a critical vulnerability: decode-node failures destroy the KV cache and force serving systems to choose between fast-but-lossy compressed loading and slow-but-exact full loading. The dominant recovery strategy treats these as the only two options, implicitly assuming that exactness and speed are fundamentally opposed. In practice, this assumption misses a structural property of the setting. An INT4 checkpoint and its FP16 original are not two independent approximations. They encode the same generation history at two levels of precision, built from identical weights and prompt history. We call this precision duality, and it enables recovery that is both immediate and exact. We present Specula, a speculative recovery framework: the INT4 checkpoint serves as a draft for immediate generation while the exact FP16 state loads in the background, and rejection sampling verifies every committed token against the exact state as it arrives. Per-channel INT4 quantization preserves the variance structure governing argmax stability, yielding 97–99% acceptance across all tested models without tuning. Across five models from 7B to 32B parameters at 8K to 64K context, Specula delivers its first retractable token at least $3.0\times$ before the strongest lossless baseline commits, sustains continuous output through the recovery window where single-draft and lossless variants deliver one token or none, and shows no measurable quality degradation on HumanEval, GSM8K, and MMLU. By reframing recovery as a precision-duality problem rather than a speed–quality trade-off, Specula suggests a new perspective on fault-tolerant LLM serving.

1 INTRODUCTION

Inference systems for large language models (LLMs) increasingly adopt disaggregated architectures that place the prefill phase and the decode phase on separate node pools Zhong et al. (2024); Patel et al. (2024); Qin et al. (2024). The prefill phase processes the input prompt in parallel to produce the key-value (KV) cache and is compute-bound, while the decode phase generates tokens autoregressively from the cached state and is memory-bandwidth-bound. Separating the two phases enables independent scaling and improves hardware utilization Zhong et al. (2024); Patel et al. (2024).

This design assumes reliable KV cache delivery between nodes. When a decode node fails due to spot instance preemption, hardware faults, or load-balancing migration, the KV cache is lost and must be reconstructed. In production clusters with hundreds of spot instances, such failures occur multiple times per hour Miao et al. (2024b), making fast recovery a first-order system requirement rather than an edge case. The FP16 KV cache grows linearly with context length and reaches 8–17 GB for 7B–32B models at 64K tokens, as shown in Figure 1(a). Production systems maintain warm standby nodes Miao et al. (2024b); Qin et al. (2024), so KV cache restoration is the dominant recovery bottleneck.

The dominant recovery strategy treats compressed and exact loading as a binary choice: fast-but-lossy versus slow-but-exact. Under this view, loading an INT4 compressed checkpoint is fast but approximate, while loading the full FP16 checkpoint is exact but slow. Figure 1(b)(c) illustrates the cost: full recomputation grows superlinearly with context while the compressed load grows linearly, and at 64K the gap between the two exceeds 40-fold. Prior work addresses this cost through warm

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.

standbys, optimized load scheduling, and checkpoint compression, but the binary framing itself has remained unchanged across the literature.

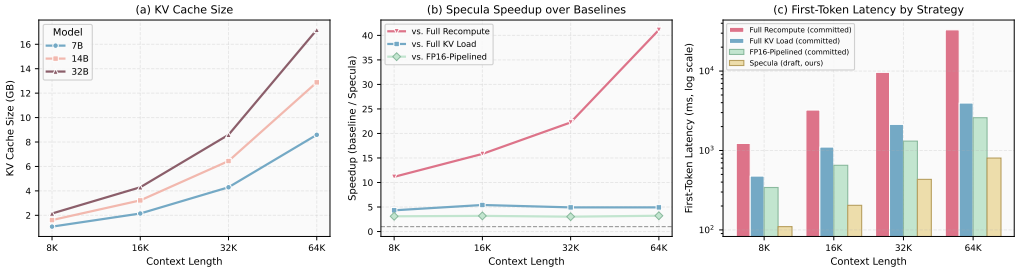

Figure 1: Empirical motivation. (a) FP16 KV cache size vs. context. (b) Specula speedup over each baseline. (c) Absolute first-token latency.

In practice, the INT4 checkpoint and the FP16 original are not independent approximations: they encode the same generation history at two levels of precision, derived from identical weights and prompts. Any divergence is strictly quantization noise, which rarely shifts the stable argmax. Treating the INT4 checkpoint as merely lossy discards this structural relationship and forecloses a recovery strategy that is both immediate and exact.

Motivated by this, we propose Specula, a speculative recovery framework exploiting precision duality. Specula uses the INT4 checkpoint as a draft: generation resumes immediately from the compressed state while the exact FP16 cache loads in the background, and rejection sampling verifies every committed token against the exact state as it arrives. Because the draft shares identical weights and history with the target, Specula yields 97–99% per-token acceptance natively, far above the 60–80% typical of auxiliary draft models. It thus delivers continuous output through the recovery window while remaining strictly lossless.

We evaluate Specula on five models (7B to 32B parameters) from the Llama, Mistral, and Qwen families across mathematical reasoning, code generation, and general knowledge tasks. This paper makes three contributions:

- We identify that the INT4 checkpoint and its FP16 original encode the same generation history at two levels of precision, not two independent approximations, and show that this structural property makes a concurrent-generation strategy both natural and viable, one that is invisible under the conventional speed–quality framing of KV-cache recovery.
- We instantiate speculative recovery in Specula, overlapping per-layer dequantization with the FP16 I/O window, adapting draft length dynamically to storage bandwidth, and implementing an INT4 checkpoint protocol that achieves $3.94\times$ compression across all tested architectures without anchor layers.
- Under a protocol with process-separated checkpoint writes, verified-cold page cache, and median-of-7 reporting, Specula reaches its first retractable token 3.0–$4.4\times$ before the strongest lossless baseline and shows no measurable quality degradation on HumanEval, GSM8K, and MMLU across all tested models.

2 Related Work

Disaggregated serving and KV compression. Continuous-batching serving systems Yu et al. (2022); Kwon et al. (2023); Agrawal et al. (2024) and disaggregated architectures Zhong et al. (2024); Patel et al. (2024) optimize throughput and phase separation but treat KV cache recovery as a secondary routing concern. Fault-tolerant serving systems like SpotServe Miao et al. (2024b), ServerlessLLM Fu et al. (2024a), and Mooncake Qin et al. (2024) address node failures by relying on expensive full recomputation (22–71 s at 64K across our models, Table 3). KV compression via weight quantization Frantar et al. (2022); Lin et al. (2024); Xiao et al. (2023) or KV-specific quantization Liu et al. (2024b); Hooper et al. (2024); Kang et al. (2024) and sparsity Tang et al. (2024); Li et al. (2024); Cai et al. (2024b); Sun et al. (2024); Liu et al. (2024a) reduces footprint by accepting permanently approximate output. Specula bridges the two: it complements serving

systems by targeting the tail-latency of cache restoration, and uses INT4 not as a permanently lossy state but as a temporary, structurally identical draft whose bounded divergence is corrected during the background I/O window.

Speculative decoding for inference acceleration. Speculative decoding was introduced as a lossless method to accelerate inference by having a draft model propose tokens that a target model verifies in parallel Leviathan et al. (2022); Chen et al. (2023). Subsequent work improves the draft proposal using learned heads Cai et al. (2024a), tree-structured verification Miao et al. (2024a), retrieval and lookahead Fu et al. (2024b), self-speculative early exit Zhang et al. (2024), branch parallelism Shen et al. (2026), online draft adaptation Liu et al. (2023), and collaborative multi-model drafting Li et al. (2026), with a broader survey of the design space in Xia et al. (2024). These methods operate in steady-state inference, treat the draft as an auxiliary model trained or selected independently of the target, and target wall-clock throughput while preserving the target distribution. Specula operates in a structurally different setting: the draft is the same model at lower precision, available only during a fault-recovery window. Precision duality eliminates the need for auxiliary training and makes high acceptance empirically robust across the evaluated models and contexts rather than the result of tuning. Rejection sampling serves as a correctness guarantee rather than a speedup mechanism, and the goal is lossless recovery with continuous output, not steady-state throughput.

3 METHOD

3.1 PRELIMINARIES

Notation. Let a transformer Vaswani et al. (2017) have N layers and H_{kv} KV heads of dimension d_h. The FP16 KV cache at sequence length L occupies

$$\mathcal{S}_{\text{FP16}} = 4NLH_{\text{kv}}d_h \text{ bytes}, \tag{1}$$

reaching 8–17 GB for 7B–32B models at 64K context ($H_{\text{kv}} < H$ under GQA Ainslie et al. (2023)). When a decode node fails, three recovery strategies are available: Full Recompute at cost $O(L^2N)$, Full KV Load at cost $O(\mathcal{S}_{\text{FP16}}/B)$, and Compressed Load at cost $O(\mathcal{S}_{\text{FP16}}/rB)$ where r is the compression ratio and B the storage bandwidth.

Precision duality. After a failure, both an INT4 compressed checkpoint and the original FP16 cache reside on storage. These are not two independent approximations. They encode the same generation history at two levels of precision, built from the same model and the same prompt. Any divergence between the token distributions they induce is caused by quantization noise alone. Speculative recovery exploits this duality to generate output immediately from the cheaper representation while the exact state loads, and to certify every committed token via rejection sampling. Because both precision views already coexist on storage, this requires no additional writes or network transfers beyond what the checkpoint protocol already stores. The INT4 read overlaps the FP16 I/O window and does not extend the critical-path latency, so the total wall-clock cost is bounded by the FP16 stream alone. No auxiliary model is needed.

3.2 DRAFT STATE

The axis choice for INT4 quantization directly determines whether speculative recovery is viable. Keys exhibit high variance across channels but low variance across tokens, while values exhibit the opposite. Matching the grouping axis to the dominant variance component preserves argmax stability: quantization noise rarely shifts the top-logit token, so the acceptance rate is structural rather than the result of tuning. Reversing the axis choice would destroy this variance alignment and sharply lower acceptance. To see why, note that per-channel quantization bounds the per-element reconstruction error by $\epsilon \leq s/2$ where s is the channel scale factor. When the logit gap between the top-1 and top-2 tokens exceeds 2ϵ after attention, the argmax is invariant to the quantization noise. Because the logit gap at greedy positions is typically much larger than the quantization error, the draft and exact states agree on the top token in the overwhelming majority of decoding steps, making high acceptance a consequence of the quantization design rather than an empirical coincidence.

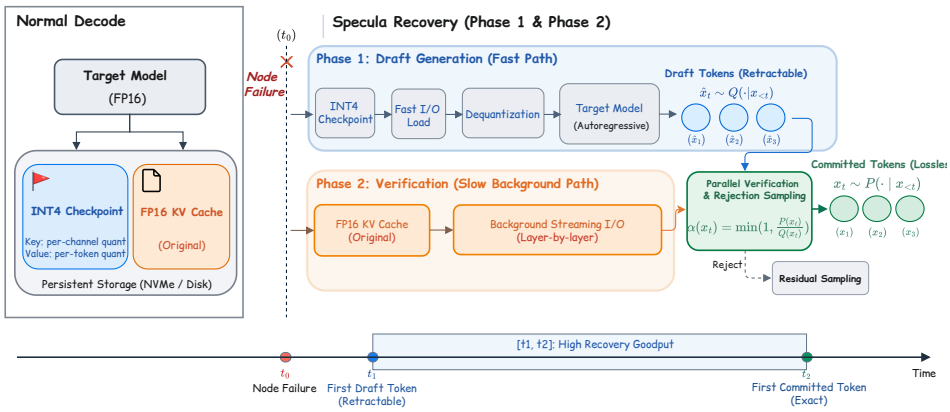

Figure 2: Overview of Specula. Left: periodic INT4 checkpointing during normal decode. Right: two-phase recovery pipeline.

Concretely, keys are quantized per-channel with scale factor $s_j = \max_i |T_{i,j}|/7$ shared across all token positions for head dimension j. Values are quantized per-token with group size g=128 along the head dimension. We use symmetric INT4 quantization throughout.

3.3 Correctness of Verification

Rejection sampling Leviathan et al. (2022); Chen et al. (2023) produces tokens whose marginal distribution is exactly the target P, regardless of the draft distribution Q. Specula uses this as a verification primitive, not a speedup mechanism: every token committed to the user has the distribution of generation from the exact FP16 state. A draft token $x_t \sim Q(\cdot \mid x_{<t})$ is accepted with probability

$$\alpha(x_t) = \min\left(1,\ \frac{P(x_t \mid x_{<t})}{Q(x_t \mid x_{<t})}\right) \tag{2}$$

If rejected, a correction is sampled from the residual $P'(x) \propto \max(0,\ P(x) - Q(x))$. Combining the accept and resample branches, the marginal distribution of every committed token is exactly the target, independent of the draft:

$$\Pr[\text{output}_t = x] = Q(x)\,\alpha(x) + \left(1 - \textstyle\sum_{x'} Q(x')\alpha(x')\right) P'(x) = P(x \mid x_{<t}). \tag{3}$$

This is the losslessness guarantee Specula relies on, with the full proof in Appendix D.

3.4 Two-Phase Recovery Pipeline

Upon a failure, the recovery node runs two concurrent phases, illustrated in Figure 2 and formalized in Appendix A.

Phase 1: Draft Generation. The node loads the INT4 checkpoint, dequantizes it on the GPU, and begins generation immediately. Each draft token $\hat{x}_t \sim Q(\cdot \mid \hat{\mathbf{K}}, \hat{\mathbf{V}}, x_{<t})$ is delivered to the user as a retractable token, yielding a rapid first-token latency (t_1). Draft length adapts to bandwidth: $N_{\text{draft}} = \lfloor (\mathcal{S}_{\text{FP16}}/B)/T_{\text{decode}} \rfloor$.

Phase 2: Verification. Concurrently, the exact FP16 cache streams layer by layer. Once loaded, all draft tokens are verified in a single parallel pass: each is accepted with probability $\min(1, P/Q)$ (an argmax match at T=0). On the first rejection the token is corrected from the residual P', subsequent drafts are discarded, and generation resumes exactly from the FP16 state, yielding the first committed token (t_2).

Delivery semantics. A retractable token may be corrected, so Specula targets serving interfaces with token-level retraction, as interactive front-ends already provide. We report three metrics from failure time t_0: t_1 (first retractable draft token), t_2 (first committed token), and Recovery Goodput (RG), accepted draft tokens divided by $t_2 - t_0$. Lossless baselines have $t_1 = t_2$ and RG = 0,

while Specula decouples them ($t_1 \ll t_2$, RG > 0). For RG's numerator we report the single-round accepted prefix in per-configuration tables and the window match count in the cross-model comparison, since the prefix is volatile to a first-token rejection. Both are recorded per trial, with detailed metric semantics in Appendix A. At a mean per-token acceptance rate of 97–99%, the expected number of retracted tokens per recovery window is under two regardless of draft length, and multi-token rollbacks require a contiguous sequence of rejections whose probability falls geometrically with length.

3.5 CHECKPOINTING

Keys are quantized per-channel and values per-token (group size 128), matching their variance structures Liu et al. (2024b); Hooper et al. (2024) and yielding $r \approx 3.94\times$ compression (Equation 6) across all architectures. Anchor layers add $<0.1\%$ acceptance rate at the cost of compression (Section 4.7) and are omitted, with full quantization formulas in Appendix A. Deployments checkpoint on preemption warnings at zero steady-state cost, or every 500–1,000 tokens for unannounced-failure coverage (Section 4.5). Prior work on fault-tolerant training Thorpe et al. (2023); Duan et al. (2024); Mohan et al. (2021) has shown that checkpoint frequency and staleness trade off smoothly. Specula inherits this flexibility on the inference side.

Lossless guarantee and complexity. Because rejection sampling reproduces the target distribution P exactly regardless of the draft Q Leviathan et al. (2022), every committed token matches generation from the exact FP16 state: bit-identical at $T{=}0$ and distributionally equal at $T{>}0$ (formal statement and proof in Appendix D). By overlapping dequantization with I/O, Specula reaches its first token in $O(LNH_{\text{kv}}d_h/rB)$, bounded only by the compressed load, versus $O(L^2N)$ for recompute and $O(LNH_{\text{kv}}d_h/B)$ for exact loading (Table 4 in Appendix A).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Setup. We evaluate five instruction-tuned models spanning 7B to 32B parameters: Llama-3-8B Grattafiori et al. (2024), Mistral-7B Jiang et al. (2023), Mistral-Nemo-12B Mistral AI & NVIDIA (2024), Qwen2.5-14B, and Qwen2.5-32B Hui et al. (2024), at 8K–64K context. Experiments run on two A800 GPUs with a local NVMe RAID volume sustaining 3.2 GiB/s effective read bandwidth. The recovery node inherits pre-loaded model weights so only the KV cache is restored. This reflects the standard disaggregated serving deployment, where model weights reside in GPU memory or host pinned memory throughout the service lifetime Miao et al. (2024b); Qin et al. (2024). The KV cache is the dynamic per-request state that is lost on failure and must be reconstructed. Each reported latency is the median of 7 cold-cache trials with strategies interleaved in shuffled order, and a speedup is reported as significant only if the two cells' trial ranges do not overlap. Tasks: GSM8K Cobbe et al. (2021) (8-shot CoT), MMLU Hendrycks et al. (2020) (5-shot, 5,700 questions), HumanEval Chen et al. (2021) (164 problems, exact match), and bit-identical verification on 20 diverse prompts.

Baselines. We compare five strategies against the $3.94\times$-compressed INT4 checkpoint (loaded cold from NVMe, matching the production workflow). *Full Recompute* (lossless) reruns prefill. *Full KV Load* (lossless) reads the FP16 checkpoint in one blocking read. *FP16-Pipelined* (lossless) overlaps per-layer host-to-device copies with the next layer's read, the strongest lossless baseline and not reported in prior work. *Dequantize-First* (lossy) loads the INT4 checkpoint and serves from the approximate state permanently. *Specula* (lossless) pipelines the INT4 load, drafts immediately, and verifies against the FP16 background stream. Within each precision pair the only difference is scheduling.

Table 1: HumanEval exact-match at $T{=}0$. Draft: INT4-only, no verification. AR: per-token acceptance rate.

Model	Draft	Specula	AR
Llama-3-8B	6.1%	100%	97.1%
Mistral-7B	15.9%	100%	98.7%
Nemo-12B	17.7%	100%	99.0%
Qwen2.5-14B	5.5%	100%	98.4%
Qwen2.5-32B	6.7%	100%	98.7%

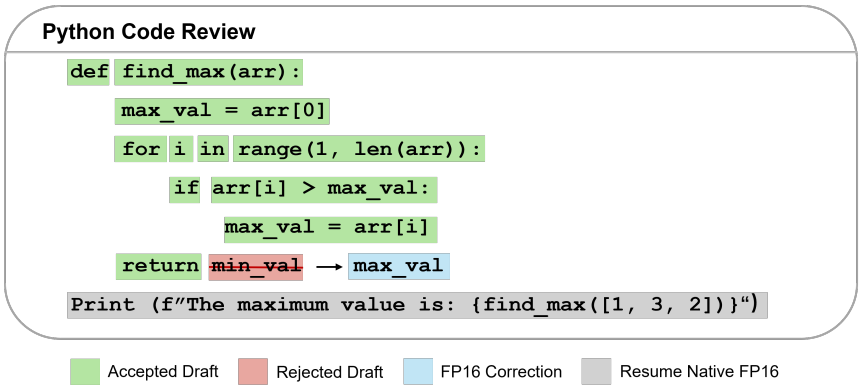

Figure 3: Token-level rejection sampling (Qwen2.5-32B, 32K, T=0). 38 draft tokens: 37 accepted (green), 1 rejected (`min_val`, red), corrected to `max_val` (blue) in <30 ms.

4.2 Quality Validation

HumanEval Exact-Match Recovery. HumanEval is the most demanding test: a single incorrect token anywhere in a function body fails the entire problem. Table 1 shows that the INT4 draft alone matches the full-precision baseline on only 5.5–17.7% of problems, yet rejection sampling recovers 100% across all five models. Nemo-12B achieves both the highest draft match (17.7%) and the highest AR (99.0%). Larger models exhibit the lowest draft match rates (5.5–6.7%) yet per-token AR stays above 97%, indicating scattered single-token errors rather than systematic drift. Figure 3 illustrates one such error and its correction on a Qwen2.5-32B example.

Table 2: GSM8K accuracy (8-shot CoT, T=0). Gap: Baseline minus Specula.

Model	N	Baseline	Draft	Spec.	Gap
Llama-3-8B	200	76.5	72.5	76.5	0
Mistral-7B	200	40.5	36.5	40.5	0
Nemo-12B	200	80.5	75.0	80.5	0
Qwen2.5-14B	200	92.0	93.5	92.0	0
Qwen2.5-32B	200	89.0	84.5	89.0	0

Pipelining the INT4 load without verification cannot substitute for Specula: the Draft baseline shares the same t_1 path yet matches the FP16 baseline on only 5.5–17.7% of HumanEval problems, because dequantization recovers a discretized approximation rather than the original activations. This is why Table 3 anchors the primary comparison to FP16-Pipelined.

GSM8K Mathematical Reasoning. Table 2 shows zero accuracy gap across all models even though INT4 alone degrades accuracy by 1.2–5.5%. Full results in Appendix B.

MMLU General Knowledge. Draft degradation below 0.7% across all models, with full results in Appendix B.

4.3 Recovery Latency

Table 3 compares recovery latency under the same hardware, storage, and model for all five strategies. The goal is not architectural parity across baselines, since they differ fundamentally in whether they produce lossless output.

Pipelining and compression contribute through separable mechanisms. Pipelining alone improves the INT4 path by 1.34–2.00× and the FP16 path by 1.23–1.78×, with overlap efficiency saturating at 0.93–0.96. The decisive lever is byte count: Specula reaches its first token 3.0–4.4× before FP16-Pipelined commits, widening toward the 3.94× compression ratio as both paths become I/O-bound. Pipelining sets a constant factor and compression sets the asymptote. The 9–47× gap over full recomputation is an upper reference only.

Bandwidth sweep. Sweeping from 3.2 to 0.35 GiB/s drives the two mechanisms in opposite directions. The pipelining gain collapses from 1.52× to 1.04× as I/O dominates, while the advantage over the lossless baseline rises from 3.6× to 3.9×, converging on the compression ratio. Specula's

Table 3: Recovery latency (ms), five models. Draft strategies report t_1, lossless report t_2. S/L: Specula t_1 / FP16-Pipelined t_2. S/R: over Full Recompute.

Model	Strategy	8K	16K	32K	64K	S/L	S/R
Llama-3-8B	Full Recompute	1,227	3,229	9,633	33,024		
	Full KV Load	476	1,104	2,132	3,955		
	FP16-Pipelined	342	651	1,312	2,586		
	Dequantize-First	192	386	758	1,469		
	Specula	**110**	**204**	**433**	**803**	3.0–3.2×	11–41×
Mistral-7B	Full Recompute	993	2,474	6,940	22,426		
	Full KV Load	463	865	2,175	4,669		
	FP16-Pipelined	312	601	1,605	2,627		
	Dequantize-First	157	282	572	1,361		
	Specula	**100**	**177**	**380**	**679**	3.1–4.2×	10–33×
Mistral-Nemo-12B	Full Recompute	1,433	3,506	9,593	30,005		
	Full KV Load	534	1,274	2,270	5,557		
	FP16-Pipelined	397	856	1,595	3,944		
	Dequantize-First	196	398	664	1,290		
	Specula	**123**	**271**	**444**	**903**	3.2–4.4×	12–33×
Qwen2.5-14B	Full Recompute	1,797	4,527	12,768	40,839		
	Full KV Load	759	1,556	3,238	6,074		
	FP16-Pipelined	606	1,063	2,227	3,875		
	Dequantize-First	271	486	835	2,041		
	Specula	**190**	**315**	**532**	**1,059**	3.2–4.2×	9–39×
Qwen2.5-32B	Full Recompute	3,878	9,072	23,984	70,635		
	Full KV Load	999	2,020	3,891	7,329		
	FP16-Pipelined	737	1,639	3,006	5,538		
	Dequantize-First	356	622	1,200	2,551		
	Specula	**232**	**451**	**895**	**1,506**	3.2–3.7×	17–47×

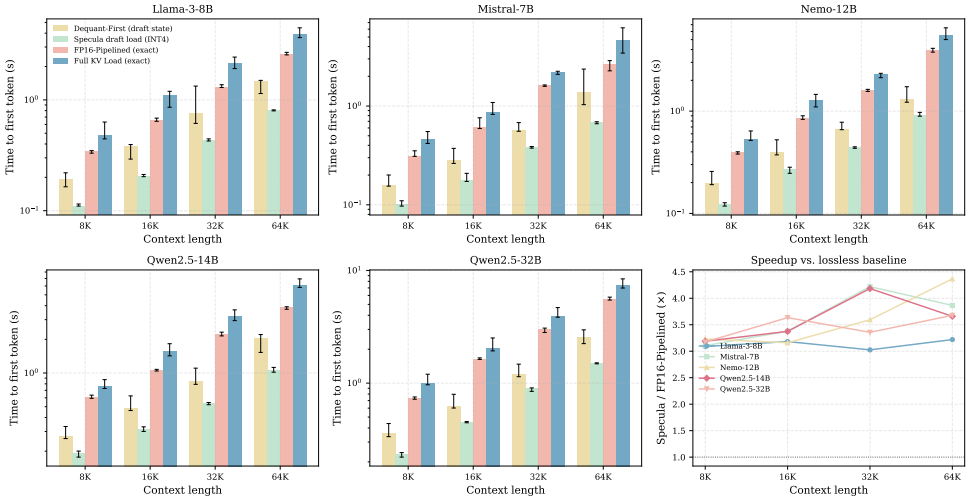

Figure 4: First-token latency, five models, log scale. Each panel shows four load strategies across context. Bottom-right: Specula/FP16-Pipelined speedup across models.

edge is largest in the bandwidth-starved regime where recovery latency hurts most, and this trend holds across all five models (Figure 10, Appendix C).

The cost of verification. Rejection sampling runs in a background stream and never enters the t_1 critical path, so the INT4-only baseline shares Specula's t_1 column in Table 3. The quality difference is visible in Tables 1 and 2: without verification, the INT4 state matches full precision on only 5.5–

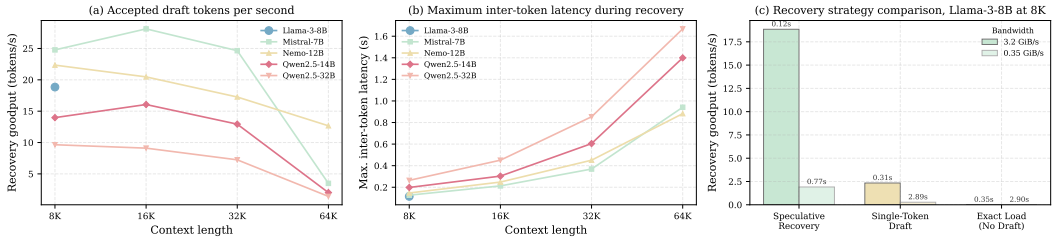

Figure 5: Recovery goodput, document prompt. (a) Within each model's native window RG rises with context. Past the native window RG collapses (three 32K-window models at 64K), while Nemo-12B (128K native) avoids this collapse and sustains high goodput throughout. Llama-3-8B is at its 8K native context only. (b) Maximum inter-token latency stays bounded by the first-token latency at every point. (c) Recovery strategy comparison for Llama-3-8B at 8K under native (3.2 GiB/s) and throttled (0.35 GiB/s) bandwidth, with inter-token gap annotated above each bar.

17.7% of HumanEval problems and deviates by 1.2–5.5 accuracy points on GSM8K. Verification closes both gaps to zero at no measurable t_1 cost.

Precision duality holds across models and contexts. Acceptance rate stays at 97.1–99.0% across all five models under greedy decoding and does not degrade with model size or context length (Appendix C), confirming that the divergence between the two precision views is bounded by quantization noise rather than task content or model scale. This stability is the mechanism behind the recovery goodput results in Section 4.4.

4.4 Recovery Goodput

A single draft token minimizes t_1, but the recovery strategies diverge sharply over the remaining load window (Figure 5(c)). The single-draft variant goes silent after its one token, with the maximum inter-token latency growing from 0.32 s at native bandwidth to 2.9 s at throttled 0.35 GiB/s. Specula holds that gap to the first-token latency itself, an order of magnitude lower at the throttled end, because continuous drafting fills exactly the silent window a single draft leaves open. The retracted tail is paid only when verification overrules a delivered token.

Extending the measurement to all five models and context lengths (Figure 5) exposes a structural dependence on the model's native attention window. Within the native window, recovery goodput stays in a 7–28 tokens/s band and rises with context as the draft matches a growing share of window tokens, so the continuous-draft benefit is a property of precision duality rather than a favorable single configuration. Past the native window the picture changes: the three models with a 32K native window are forced to extrapolate at 64K and their matched-window count collapses, while Nemo-12B, whose native window is 128K, avoids this collapse and sustains goodput above 12 tokens/s at 64K, confirming the drop is an attention-window artifact rather than a limit of the recovery mechanism. Even in the collapsed regime Specula sustains a continuous stream where single-draft and lossless variants deliver one token or none, and the maximum inter-token latency remains bounded by the first-token latency at every point. This comparison uses the window match count rather than the single-round prefix, which a rare first-token rejection can zero out despite the rest of the window matching.

4.5 Unannounced Failures

To cover failures with no checkpoint boundary, a serving process writes an INT4 plus FP16 pair every 500 or 1,000 tokens, logs each token id to an fsynced journal, and is killed with SIGKILL at varying offsets. Recovery loads the newest checkpoint and replays the logged suffix in one forward pass. Suffix replay scales at ≈0.2 ms per token and stays a small fraction of total recovery time: the draft path reaches its first token in 136–256 ms and the exact path commits in 346–472 ms, so recovery cost is governed by checkpoint loading, not replay. The continuation matches a full-recompute reference for 64 of 64 greedy tokens in half the scenarios. Divergence in the rest is a floating-point artifact of chunk boundaries, not lost state.

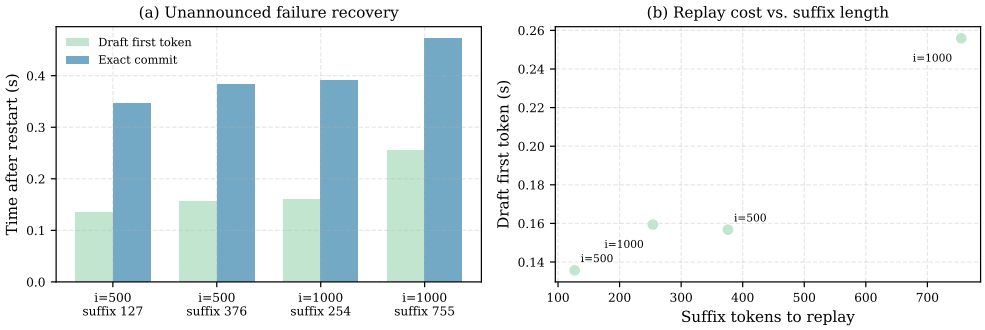

Figure 6: Recovery from unannounced SIGKILL, median of 5 cold trials. (a) Draft first token and exact commit per checkpoint interval i and suffix length. (b) Suffix replay scales at $\sim$0.2 ms/token.

Checkpoint write overhead. A synchronous dual-state write at 6K–7.6K context stalls decode for 2.0–2.5 s, amortizing to a 5–6% throughput tax at a 1,000-token interval and less with overlap. Holding both states costs $1.25\times$ the FP16 footprint regardless of interval. Deployments with preemption warnings sidestep the tax entirely by checkpointing reactively.

4.6 Scaling with Context Length

Longer contexts favor Specula on both comparisons, but for different reasons, as Table 3 shows. The 8K–64K range spans the full practical context lengths seen in today's disaggregated deployments. Against the strongest lossless baseline the speedup grows with context, from around $3.2\times$ at 8K toward the $3.94\times$ compression ratio at 64K, as both paths saturate I/O and byte count becomes the sole differentiator. Against full recomputation the growth is superlinear, from $9\times$ to $47\times$, because the two sides scale in different complexity classes: recomputation is $O(L^2N)$ while checkpoint I/O is $O(LN/r)$. The first curve is bounded by compression and the second is unbounded in L, so the regime that most stresses recovery, long context, is exactly where Specula's margin is largest.

4.7 Ablation Study

Effect of anchor layers. This ablation tests the premise that per-channel INT4 quantization without anchor layers is optimal under precision duality. If acceptance rate were sensitive to anchor layers, the compression ratio would need to be sacrificed to maintain draft quality, and the $3.94\times$ speedup claim would not hold in general. Anchor layers retain selected layers in FP16 to improve draft quality at the cost of compression. Sweeping from zero anchors through 1+1, 2+2, and 4+4 configurations leaves AR unchanged at 98.4% while compression degrades from $3.94\times$ at zero anchors to $3.10\times$ at 4+4, confirming zero anchors is optimal across all five models (Figure 9, Appendix C). AR is stable across decoding temperatures (98.2% at T=0, 99.4% at T=0.6) and degrades less than 2% over 100 autoregressive steps.

5 Conclusion

We presented Specula, a speculative recovery framework for disaggregated LLM serving. The key insight is that an INT4 checkpoint and its FP16 original encode the same generation history at two levels of precision, not two independent approximations. Specula uses the compressed checkpoint as a draft for immediate generation while the exact state loads in the background, certifying every committed token via rejection sampling. Across five models at 8K–64K context, this delivers a first retractable token 3.0–$4.4\times$ before the strongest lossless baseline with no measurable quality degradation. Precision duality is not limited to INT4: any two representations of the same inference state differing only in fidelity admit the same treatment, making speculative recovery a general pattern for fault-tolerant LLM serving.

AI USE STATEMENT

We used generative AI tools (Claude) to assist with proving mathematical claims and editing draft text for clarity. We did not use generative AI tools to generate core technical contributions, experimental design, or algorithmic implementations. All AI-generated content was manually reviewed and verified against original sources and experimental data by the authors. We take full responsibility for the final content of this work, including all claims, code, and experimental results.

ETHICS STATEMENT

This work focuses on improving fault tolerance in LLM serving systems. The techniques presented could benefit both commercial and open-source deployments. We do not foresee negative societal impacts beyond those inherent to large language models themselves. Our experiments were conducted on isolated research infrastructure and did not involve human subjects, user data collection, or deployment to production systems serving real users.

REPRODUCIBILITY STATEMENT

We provide detailed experimental setup in Section 4.1, including configurations and protocols. All timing measurements follow a cold-cache protocol (Section 4.1) with process separation between checkpoint writers and readers. We report median latencies over seven trials and specify variance where relevant. Code for the Specula prototype, including checkpoint generation, dequantization pipeline, and speculative verification logic, will be made available upon publication. Benchmark datasets are publicly available, and we specify exact prompting formats in the appendix.

REFERENCES

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, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. 2024a.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *ArXiv*, abs/2406.02069, 2024b.

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.

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

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

Jiangfei Duan, Ziang Song, Xupeng Miao, Xiaoli Xi, Dahua Lin, Harry Xu, Minjia Zhang, and Zhihao Jia. Parcae: Proactive,liveput-optimized dnn training on preemptible instances. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 1121–1139, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. 2022.

Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. Serverlessllm: Low-latency serverless inference for large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 135–153, 2024a.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. 2020.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *ArXiv*, abs/2310.06825, 2023.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*, 2024.

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.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. volume 1, 2022.

Jinze Li, Yixing Xu, Guanchen Li, Jinfeng Xu, Shuo Yang, Yang Zhang, Xuanwu Yin, Dong Li, Edith CH Ngai, and Emad Barsoum. Beyond the target: From imitation to collaboration in speculative decoding. *arXiv preprint arXiv:2605.24793*, 2026.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. volume 6, pp. 87–100, 2024.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024a.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. 2024b.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024a.

Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. Spotserve: Serving generative large language models on preemptible instances. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1112–1127, 2024b.

Mistral AI and NVIDIA. Mistral NeMo. `https://mistral.ai/news/mistral-nemo/`, 2024.

Jayashree Mohan, Amar Phanishayee, and Vijay Chidambaram. Checkfreq: Frequent, fine-grained dnn checkpointing. In *19th USENIX Conference on File and Storage Technologies (FAST 21)*, pp. 203–216, 2021.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132, 2024.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Heyi Tang, Feng Ren, Teng Ma, Shangming Cai, Yineng Zhang, Mingxing Zhang, et al. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *ACM Transactions on Storage*, 2024.

Yuhao Shen, Junyi Shen, Quan Kong, Tianyu Liu, Yao Lu, and Cong Wang. Specbranch: Speculative decoding via hybrid drafting and rollback-aware branch parallelism. In *International Conference on Learning Representations*, volume 2026, pp. 63063–63100, 2026.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. Bamboo: Making preemptible instances resilient for affordable training of large dnns. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 497–513, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 30, 2017.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099, 2023.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX symposium on operating systems design and implementation (OSDI 22)*, pp. 521–538, 2022.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. pp. 11263–11282, 2024.

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, 2024.

A Implementation Details

Algorithm 1 formalizes the two-phase speculative recovery pipeline described in Section 3.4.

Algorithm 1 Speculative Recovery Pipeline of Specula

Require: Compressed draft KV state D, pointer to full-precision KV state F, target model $\mathcal{M}$, temperature T

1: Load D from storage; dequantize layer-by-layer on GPU: $\hat{\mathbf{K}}^{(\ell)}, \hat{\mathbf{V}}^{(\ell)} \leftarrow \mathrm{Dequantize}(D^{(\ell)})$ for $\ell = 1, \ldots, N$
2: Initialize asynchronous layer-by-layer streaming of exact state $\mathbf{K}^*, \mathbf{V}^*$ from F
3: $t \leftarrow 0$
4: **while** streaming of $\mathbf{K}^*, \mathbf{V}^*$ is incomplete **do**
5: Sample draft token $\hat{x}_t \sim Q(\cdot \mid \hat{\mathbf{K}}, \hat{\mathbf{V}}, x_{<t})$ using model $\mathcal{M}$
6: Append $\hat{x}_t$'s KV to draft cache
7: Output $\hat{x}_t$ to user {Phase 1: draft generation}
8: Cache draft probabilities $q_t \leftarrow \mathrm{softmax}(\mathrm{logits}_t)$
9: $t \leftarrow t + 1$
10: **end while**
11: {Phase 2: parallel verification}
12: Compute all target probabilities $\{p_i\}_{i=0}^{t-1}$ via a single parallel forward pass over $(\mathbf{K}^*, \mathbf{V}^*, \hat{x}_{0:t-1})$
13: Truncate exact KV cache to prompt length (discard draft token entries)
14: **for** each generated draft token $\hat{x}_i$, $i = 0, \ldots, t-1$ **do**
15: Sample $u \sim \mathcal{U}[0, 1]$
16: **if** $T = 0$ **then**
17: Accept $\hat{x}_i$ if $\arg\max\ p_i = \hat{x}_i$
18: **else**
19: Accept $\hat{x}_i$ if $u \leq \min(1,\ p_i(\hat{x}_i)/q_i(\hat{x}_i))$
20: **end if**
21: **if** $\hat{x}_i$ is rejected **then**
22: **if** $T = 0$ **then**
23: Replace $\hat{x}_i$ with $\arg\max\ p_i$
24: **else**
25: Resample from residual distribution $P'(x) = (p_i(x) - q_i(x))^+ / \sum_{x'} (p_i(x') - q_i(x'))^+$
26: **end if**
27: Discard all draft tokens after position i
28: **break**
29: **end if**
30: Append accepted token's exact KV to $\mathbf{K}^*, \mathbf{V}^*$
31: **end for**
32: Resume generation using exact state $\mathbf{K}^*, \mathbf{V}^*$

Implementation. The full-precision KV cache is streamed layer by layer as specified in line 2 of Algorithm 1, where each chunk corresponds to the key and value tensors of one transformer layer, of size $2LH_{\mathrm{kv}}d_h \cdot 2$ bytes. For Llama-3-8B at 32K context, each layer chunk is approximately 128 MB, and the full 32-layer transfer completes in about 1.2 s at the 3.2 GiB/s NVMe bandwidth of our testbed. The number of draft tokens generated during Phase 1 is determined dynamically by the I/O transfer time: $N_{\text{draft}} = \lfloor (\mathcal{S}_{\text{FP16}}/B)/T_{\text{decode}} \rfloor$, where B is the storage bandwidth and T_{decode} is the per-token decode latency. The system naturally adapts to the available bandwidth without any bandwidth-dependent heuristic. Draft logits are cached during Phase 1 at line 8 to avoid recomputation during verification. Per-trial I/O and GPU-work accounting confirms the overlap mechanism: pipelined strategies land within 7% of the maximum of the two components rather than their sum, and per-layer GPU work is roughly 40% of I/O time on the INT4 path, bounding the pipelining gain at the observed $1.5\times$.

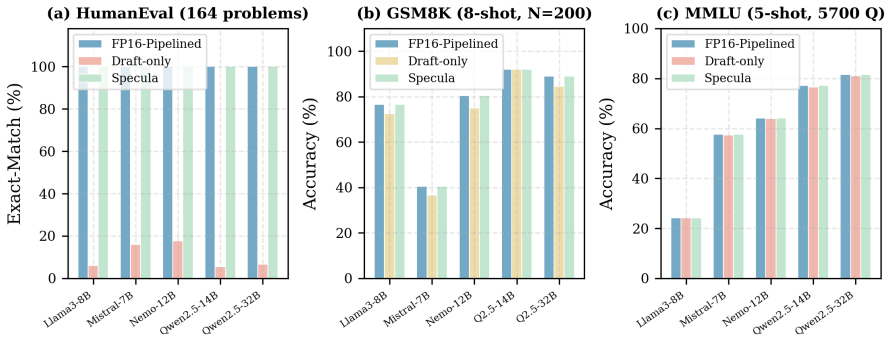

Figure 7: Empirical validation of output exactness. (a) HumanEval exact-match rates: FP16-Pipelined and Specula both reach 100% exact match, while INT4 draft-only produces correct output for only 5.5–17.7% of problems. (b) GSM8K accuracy with 8-shot chain-of-thought: Specula matches FP16-Pipelined exactly, with zero gap across all five models. (c) MMLU accuracy with 5-shot prompting: Specula matches FP16-Pipelined across all models, while the draft-only gap stays below 0.7%.

Quantization details. Keys are quantized per-channel with a single scale factor shared across all token positions for each head dimension:

$$s^K_{h,d} = \frac{\max_t |\mathbf{K}_{t,h,d}|}{7}, \quad \hat{\mathbf{K}}_{t,h,d} = \text{clamp}\left(\left\lfloor \frac{\mathbf{K}_{t,h,d}}{s^K_{h,d}} \right\rceil, -8, 7\right) \tag{4}$$

Values are quantized per-token with group size g=128 along the head dimension:

$$s^V_{t,i} = \frac{\max_{d \in G_i} |\mathbf{V}_{t,d}|}{7} \tag{5}$$

Each 4-bit integer is packed two per byte. The total compressed size is approximately $NLH_{\text{kv}}d_h$ bytes plus scale factors. Given the FP16 baseline in Equation 1, the compression ratio is:

$$r = \frac{4NLH_{\text{kv}}d_h}{NLH_{\text{kv}}d_h + \text{scales}} \approx 3.94\times \tag{6}$$

Table 4: Theoretical latency complexity of each recovery strategy. L: sequence length, N: layers, H_{kv}: KV heads, d_h: head dimension, B: storage bandwidth, B_{mem}: GPU memory bandwidth, r: compression ratio. Specula removes the dequantization term $O(LNH_{\text{kv}}d_h)$ from the critical path by overlapping it with I/O.

Strategy	**Latency Complexity**
Full Recompute	$O(L^2N)$
Full KV Load	$O(LNH_{\text{kv}}d_h/B)$
FP16-Pipelined	$O(LNH_{\text{kv}}d_h/B)$
Dequantize-First	$O(LNH_{\text{kv}}d_h/rB + LNH_{\text{kv}}d_h/B_{\text{mem}})$
Specula	$O(LNH_{\text{kv}}d_h/rB)$

B Verification of Lossless Generation

This appendix provides additional empirical validation of the lossless guarantee established in Theorem 1, supplementing the HumanEval and GSM8K results in Section 4.2. Figure 7 summarizes the results across all three benchmarks.

Bit-identical verification at T=0. For each model, 20 diverse prompts are processed using both the full-precision baseline pipeline and the Specula pipeline with greedy decoding. All five models achieve 100% sequence match and 100% token match.

Distributional equivalence at $T>0$. At non-zero temperature, rejection sampling produces samples from the target distribution rather than producing bit-identical output. This property is verified on Mistral-7B with GSM8K at $T=0.6$.

Table 5: Distributional equivalence verification on Mistral-7B with GSM8K ($N=200$). The accuracy of Specula at $T=0.6$ matches that of the full-precision baseline at the same temperature, confirming Theorem 1.

Condition	Accuracy	Correct
Exact greedy ($T=0$)	40.5%	81/200
Exact stochastic ($T=0.6$)	37.0%	74/200
Specula with RS ($T=0.6$)	37.0%	74/200

As reported in Table 5, the full-precision baseline and Specula at $T=0.6$ produce identical accuracy of 37.0%, confirming distributional equivalence. The 3.5% drop from $T=0$ reflects the known instability of chain-of-thought reasoning under stochastic decoding Cobbe et al. (2021), not information loss from the proposed method.

C Sensitivity Analysis and System Overheads

This appendix details the key design choices in Specula and their measured costs.

Acceptance rate across models and contexts. Figure 8 shows the 97–99% per-token acceptance is stable across all five models and context lengths, confirming it is a property of precision duality rather than a favorable model or prompt.

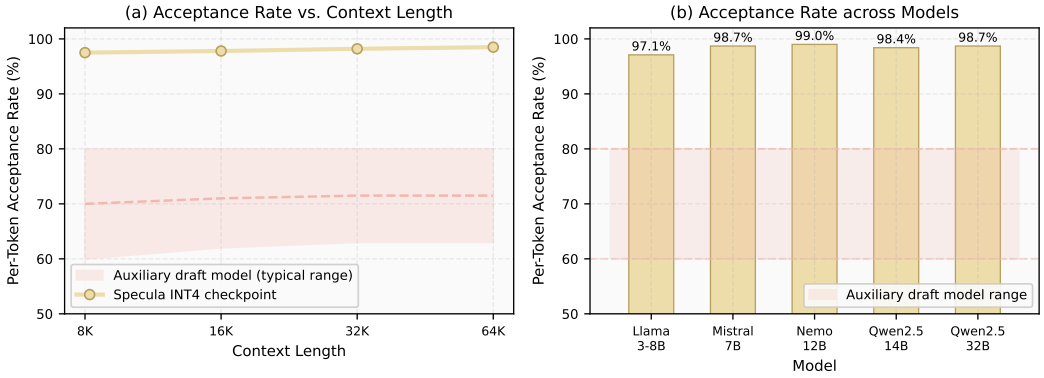

Figure 8: Precision duality. (a) INT4-checkpoint acceptance vs. the typical auxiliary-draft range, across context. (b) Acceptance across all five models, all exceeding 97%.

Cross-model anchor layer ablation. Figure 9 extends the anchor ablation to all five models: acceptance is insensitive to anchor layers while compression degrades monotonically with anchor count, so zero anchors is Pareto-optimal across the board.

Bandwidth sensitivity. Figure 10 reports the full bandwidth sweep across all five models, complementing the Llama-3-8B analysis of Section 4.3. We make no claim that loading the FP16 checkpoint beats recomputation at low bandwidth. The case for a compressed checkpoint rests on byte count, and the advantage over the strongest lossless baseline converges on the compression ratio as bandwidth falls.

Repeated failure resilience. Under a worst-case scenario of failures every 50 tokens, the cumulative overhead across 10 recoveries is 449 ms, corresponding to 4.3% of total generation time. At

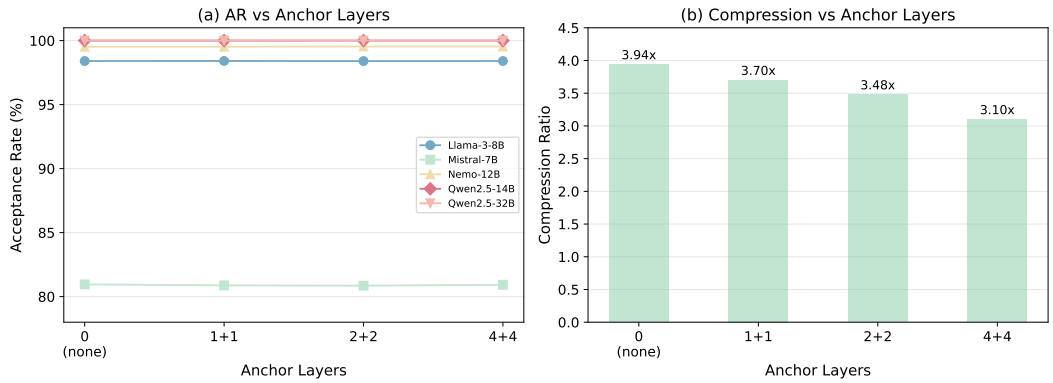

Figure 9: Cross-model anchor layer ablation. (a) Acceptance rate remains stable across 0–4 anchor layers for all models. (b) Compression ratio decreases with more anchors, confirming that zero anchors is optimal.

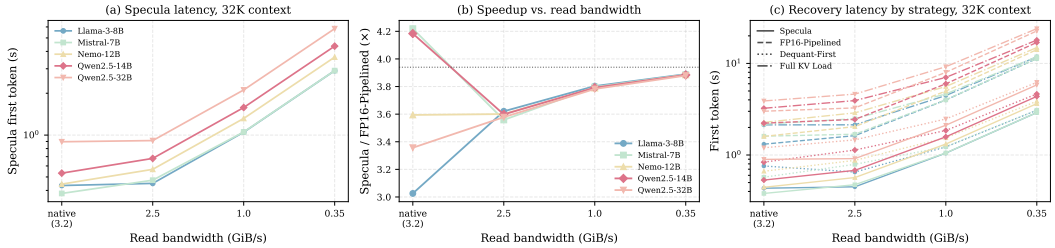

Figure 10: Bandwidth sweep at 32K, five models. (a) Specula first-token latency. (b) Specula/FP16-Pipelined speedup converges on the 3.94× compression ratio as bandwidth falls. (c) Absolute latency for all four strategies.

more realistic failure intervals of every 200 tokens, the overhead drops to 1.2%. The sublinear scaling, where 10 failures add only 4.3%, not 10× a single-failure cost, arises because each recovery reuses the already-loaded model weights and only restores the KV cache.

D Proof of Theorem 1: Lossless Recovery

Theorem 1 (Lossless recovery guarantee)**.** *Let $Q(\cdot \mid x_{<t})$ and $P(\cdot \mid x_{<t})$ denote the token distributions under the INT4-quantized and FP16 KV caches, respectively. The rejection sampling procedure in Algorithm 1 produces tokens distributed exactly according to P: at $T{=}0$ the output is bit-identical to generation from the exact state under deterministic floating-point execution, and at $T{>}0$ the output distribution equals P.*

Proof. We proceed by analyzing the procedure at a single decoding step t, and then extend the guarantee to the full generated sequence.

Single-step exactness ($T > 0$). A draft token $\hat{x}_t$ is sampled from $Q(\cdot \mid x_{<t})$ and accepted with probability $\alpha(\hat{x}_t) = \min(1, P(\hat{x}_t)/Q(\hat{x}_t))$ as defined in Equation 2. If rejected, a corrected token is drawn from the residual distribution $P'(x) = \max(0, P(x) - Q(x))/Z$, where the normalization constant is $Z = \sum_{x'} \max(0, P(x') - Q(x'))$.

For any token x in the vocabulary $\mathcal{V}$, the probability that x is the final output of this step is:

$$\Pr[\text{output} = x] = \underbrace{Q(x) \cdot \alpha(x)}_{\text{accepted directly}} + \underbrace{\left(1 - \sum_{x'} Q(x')\alpha(x')\right) \cdot P'(x)}_{\text{resampled after rejection}} \tag{7}$$

The overall acceptance probability A across the vocabulary is:

$$A = \sum_{x'} Q(x') \cdot \min\left(1, \frac{P(x')}{Q(x')}\right) = \sum_{x'} \min(Q(x'), P(x')) \tag{8}$$

The rejection probability is therefore $1-A$. The normalization constant Z of the residual distribution evaluates to:

$$Z = \sum_{x'} \max(0, P(x') - Q(x')) = \sum_{x':P(x')>Q(x')} (P(x') - Q(x')) = 1 - A \tag{9}$$

where the last equality follows from the fact that both P and Q sum to 1 ($\sum P(x') = \sum Q(x') = 1 \implies \sum_{P>Q}(P-Q) = \sum_{P\leq Q}(Q-P)$).

Substituting A and Z into Equation 7:

$$\begin{aligned} \Pr[\text{output} = x] &= \min(Q(x), P(x)) + (1-A) \cdot \frac{\max(0, P(x) - Q(x))}{1-A} \\ &= \min(Q(x), P(x)) + \max(0, P(x) - Q(x)) \\ &= P(x) \end{aligned} \tag{10}$$

The last equality holds because for any real numbers $a, b \geq 0$, $\min(a,b) + \max(0, b-a) = b$. This establishes that the marginal distribution of the output token is exactly $P(\cdot \mid x_{<t})$.

Greedy case ($T = 0$). At T=0, both P and Q are deterministic distributions placing all mass on their respective argmax tokens: let $x^* = \arg\max P$ and $\hat{x} = \arg\max Q$. If $x^* = \hat{x}$, the draft token is accepted because the verification check passes. If $x^* \neq \hat{x}$, the draft token is rejected and deterministically replaced with x^*. In both cases, the final output is x^*, which is bit-identical to the generation from the exact FP16 state.

Extension to the full sequence. This single-step exactness holds independently of the specific draft distribution Q. After verification completes at step t (whether via acceptance or correction), Algorithm 1 resumes generation strictly from the exact FP16 KV cache. Therefore, subsequent tokens are generated directly from the true distribution P. By induction over decoding steps, the full output sequence retains the exact distribution as if it were generated entirely from the exact state. □