Preprint - measurement report, single-machine study. Not peer-reviewed. Prepared in IEEE conference layout for readability; no venue affiliation implied.

Speculation Needs Rollback: A Bounded Snapshot Ring for Block-Speculative Decoding on Recurrent-Hybrid LLMs, and When It Pays

Sebastian Haas
Independent researcher · kontakt@sebastianhaas.eu

***Abstract*- Block-diffusion drafters (DFlash) draft a whole token block in one forward pass and lead current speculative-decoding rankings, but their verify step assumes the target can rewind to an arbitrary position inside the block. Kv-cache targets rewind for free; the linear-attention state of recurrent-hybrid targets (e.g. gated-DeltaNet layers in Qwen3.6/3.8) is a lossy running summary that cannot be rewound at all. We describe the *RS-ring*, a bounded per-sequence snapshot ring for recurrent state that makes partial rollback exact within a window of *K* tokens at a cost of *K*+1 state copies, and report field measurements from porting DFlash v1/v2 to two retired 16 GB gfx906 GPUs behind a llama.cpp-based fork. End to end, a 27B dense hybrid reaches 55.3 tokens/s single-stream in a production server (1.9× over its non-speculative baseline; MTP reached 1.46×, EAGLE-3 lost ground at 0.94×). The gains, however, are strongly non-uniform: they decompose by *content class*-code, prose, and reasoning ("thinking") traces-and by *context depth*. Two mechanisms carry the decomposition: acceptance *rate* varies little across classes while accepted draft *length* varies by 1.5×, and drafter cost grows with depth unless its trained sliding window is honored, which also restores acceptance (46 → 57 %). A probe that shortens drafts at depth shows verify-row count is *not* the residual cost driver. We condense the matrix into a two-input deployment gate (content class × depth) and disclose our measurement-integrity protocol, including a two-agent contention incident that briefly masqueraded as thermal throttling.**

***Index Terms*- speculative decoding, block diffusion, linear attention, gated DeltaNet, hybrid architectures, rollback, llama.cpp, GPU inference, gfx906.**

I. Introduction

Speculative decoding buys throughput by letting a cheap drafter propose tokens that the target model verifies in one batched forward pass [1], [2]. The economics are governed by two quantities: how many proposed tokens survive verification, and what the verify step costs. A third, quieter assumption underlies every implementation: *the target must be able to pretend the rejected tokens never happened.*

For pure-attention targets that assumption is trivial-discarding kv-cache rows past the accepted position is exact. It stops being trivial the moment the target contains recurrent layers. Linear-attention blocks such as gated DeltaNet [10] carry a fixed-size state that summarizes the whole prefix; advancing it is destructive. A verify batch that advances the state by eight tokens and then accepts three cannot subtract the other five. Hybrid models that interleave a few full-attention layers with many recurrent ones-the design of Qwen3.6-27B, Qwen3.8-27B, and Qwen3.6-35B-A3B [8]-therefore break the rollback assumption exactly where speculation would help most: large models on memory-bound hardware.

Block-diffusion drafters sharpen the problem. DFlash [7] drafts an entire block (8–16 tokens) per drafter call and reaches acceptance lengths far beyond classic drafters, so the target routinely needs to rewind to an *arbitrary* position inside a block, many times per second. Depth-1 or depth-2 schemes can sometimes hide the problem by re-decoding; block schemes cannot.

This paper makes three contributions. **(i)** We describe the RS-ring, a bounded snapshot ring that makes recurrent-state rollback exact within a window of *K* tokens, sized so that the ring edge coincides with the drafter's maximum block, and cheap enough (≈150 MiB per snapshot on a 27B hybrid) to run on 16 GB consumer parts. **(ii)** We report end-to-end measurements of DFlash v1 and v2 on three hybrid targets served from retired AMD gfx906 GPUs, including a three-way drafter ranking against MTP [3], [4] and EAGLE-3 [6] on identical hardware. **(iii)** We show that the headline speedup is an average over regimes that deployment should not average over: gains split by content class and context depth, and both splits have measurable mechanisms-accepted draft *length* (not rate) across classes, and windowed drafter cost across depth. A two-input gate summarizes the resulting policy.

II. Background and Problem

A. Block-diffusion drafting

DFlash [7] replaces the autoregressive drafter with a small bidirectional model that fills a block of mask tokens in one pass, conditioned on hidden states extracted from a handful of target layers and injected into the drafter's cache. Version 1 (block 16) samples each position from drafter logits; version 2 (block 8) adds two-tap dynamic convolutions and a trained *selector* lattice that scores continuation confidence directly, gating draft length via a threshold `p_min`. Both versions are trained per target; the drafter has no language-model head of its own and borrows the target's output projection.

B. Why hybrids cannot rewind

A gated-DeltaNet layer maintains state S_t updated as a data-dependent low-rank modification of S_{t-1}. The update is contractive but not invertible: S_{t-1} is not recoverable from

S_t. A verify batch of n+1 rows advances the state n+1 steps; on acceptance of $a < n$ drafts, the serving loop must resume from position a+1. Without extra machinery the only exact option is re-prefilling the sequence-which erases the speedup speculation was meant to buy.

III. The RS-Ring

The mechanism is deliberately plain (Fig. 1). Each recurrent layer keeps, per sequence, a ring of K+1 state snapshots. During a verify batch the kernel writes the post-token state of each of the last K+1 positions into consecutive ring slots; a per-sequence freshness counter (`rs_valid`) records how many trailing slots are current. A rollback to r tokens back ($1 \le r \le K$) is then a pointer move to slot $-r$, valid iff $r \le$ `rs_valid`. Requests beyond the window fail loudly and the caller falls back to non-speculative decoding for that step.

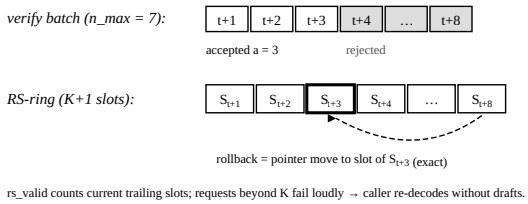

Fig. 1. The RS-ring. Each recurrent layer stores the post-token state of the last K+1 positions per sequence. Accepting a of n drafts rewinds by a pointer move; no state arithmetic, no re-prefill.

Sizing. The verify micro-batch of a block drafter holds $n_{\max}$+1 rows, so the worst-case rollback is exactly $n_{\max}$ tokens; $K = n_{\max}$ is both necessary and sufficient, and our port asserts this edge (a block-16 drafter with $n_{\max}$=15 lands exactly on it). Memory is the real price: one snapshot of the 27B hybrid's recurrent state is ≈150 MiB, so K=7 costs ≈1.2 GiB and K=15 ≈2.4 GiB per sequence; a serving default of four slots multiplies this into an out-of-memory (we measured 4.8 GiB rejected on a 16 GB card) and must be capped at the slot count actually used. The A3B MoE hybrid's state is an order of magnitude smaller and rings freely.

Integration. The ring lives in the recurrent-memory manager; the gated-DeltaNet kernels take a compile-time snapshot parameter and are byte-identical to the non-speculative path when the ring is disabled (K=0 default). Rollback support is whitelisted per architecture. One deployment rule follows from the drafter's borrowed LM head: the drafter must be placed on the device holding the target's output projection, or graph scheduling fails at load; and with all drafter layers windowed, cache setters must tolerate buffer-less inputs (the "zero dense layers" case).

IV. Experimental Setup

Hardware. Two AMD Radeon Pro VII (gfx906, 16 GB HBM2 each, power-capped 190 W), a retired 2020 workstation part. All serving is greedy (temperature 0). The fork extends llama.cpp [9] with gfx906-tuned kernels; DFlash v2 comes from its in-review llama.cpp pull request, v1 from the merged path.

Models. Targets: Qwen3.8-27B (dense hybrid, 64 layers, community abliterated variant, Q4_0), Qwen3.6-27B (dense hybrid with native MTP head, Q4_0), Qwen3.6-35B-A3B (MoE hybrid, experts 0–7 host-offloaded, Q4_0). Drafters: Qwen3.8-27B-DFlash2 (v2, block 8, selector), Qwen3.6-27B-DFlash and Qwen3.6-35B-A3B-DFlash (v1, block 16), all Q4_0-requantized after a quantization-insensitivity check (§V-D).

Method. Cooling gate <50 °C before every run; the first response after model load is discarded (≈10 % slow, reproduced 50.3 → 55.x); repeated prompts per cell; identical prompt set across arms (a code-generation task, a five-paragraph explanation task, and a long-context continuation over a 6.3k/14.9k-token code base). Decode throughput (`tg`) is reported where the server exposes timings; production numbers are wall-clock and stated as such. All GPU work ran under an advisory file lock after an incident described in §VII.

V. Results

A. Drafter ranking and end-to-end gain

On identical hardware and target (Qwen3.8-27B), the three drafter families separate decisively (Table I). DFlash v2 with the RS-ring reaches mean accepted block length τ = 6.17 and 1.76–1.91× end-to-end; the shipped MTP head manages τ ≈ 2 and 1.46×; EAGLE-3, ported and measured under the same protocol, *loses* throughput (0.94×)-its per-step drafter cost exceeds its acceptance yield on this memory-bound part. In the production server the 27B moved from 46.2 (MTP depth-2) to 55.3 tokens/s wall on code (arbitrated re-measurement: n=8 runs, median 55.2, spread 0.2 %, thermally stable to 60 °C).

TABLE I
Drafter families on Qwen3.8-27B, gfx906, greedy, short context

Drafter	τ	Speedup	Verdict
DFlash v2 (block 8) + RS-ring	**6.17**	**1.76–1.91×**	ships
MTP head, depth 2	≈2	1.46×	prose/parallel
EAGLE-3	≈3	0.94×	net loss

B. Content classes: length, not rate

Averaging over content hides the deployment-relevant structure. Table II shows the same server, same drafter, three content classes. The acceptance *rate* barely moves between reasoning traces and plain prose (76 vs. 71 %); throughput moves by 1.5×. The carrier is accepted draft *length*: reasoning output is locally predictable for long stretches, so blocks run to full depth, while free prose truncates them early (under v2's selector, via `p_min`; under v1, via first mismatch). Code sits between and is barely affected by the reasoning toggle. We initially mis-referenced a 50 tokens/s "prose" figure for the 3.6er drafter-it was measured on reasoning-mode output; re-measured with reasoning disabled it is 33.7. Reasoning traces are a *third content class*, and the fastest one; benchmark reports that do not label the class are easy to misread, including our own.

TABLE II
Content classes, Qwen3.6-27B + v1 drafter (block 16), llama-server, tg

Class	tg (tok/s)	acc. rate	mechanism
reasoning (“thinking”)	**49.8**	76 %	long runs
code	47.4	72 %	-
prose (reasoning off)	33.7	71 %	early truncation

C. Depth: honor the trained window

Our first port disabled the drafter’s sliding window (2048) to work around the zero-dense-layer crash. The consequence surfaced only at depth (Fig. 2): at a 6.3k-token prefix, drafted decode fell *below* the non-speculative baseline (25.7 vs. 29.4 tg) and acceptance collapsed to 46 %. Restoring the window recovered both at once-32.7 tg (+10 % over baseline) and 57 % acceptance-for two distinct reasons: drafter cost stops growing linearly with depth, and the drafter again sees the attention geometry it was *trained* with. Short-context behavior is bit-identical (same acceptance pattern), as it must be when the window exceeds the context.

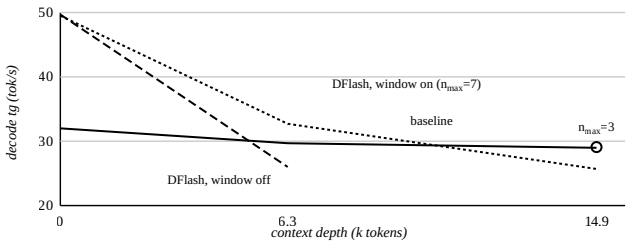

Fig. 2. Depth economics, Qwen3.8-27B + DFlash v2. Disabling the trained window makes drafting a net loss beyond ≈4k. Restoring it moves the break-even to ≈10–14k; at 14.9k, shortening drafts to n_{max}=3 reaches parity (72 % acceptance).

The residual decline is *not* verify-row count: cutting n_{max} from 7 to 3 at 14.9k raises acceptance to 72 % yet only reaches parity-the floor is the constant drafter step measured against a baseline that itself slows with depth. Two corollaries: depth-adaptive draft length is the correct serving policy, and window size is a hard datum-the 35B’s v1 drafter ships a 4096 window, which at 6k depth prunes little; combined with MoE verify rows through host-offloaded experts it stays a net loss at depth (41 vs. 61 tg) even with the window honored.

D. Transfer and quantization

Two negative results save others time. **Transfer fails:** the 3.8er’s v2 drafter attached to Qwen3.6-27B-identical hidden size, vocabulary, and layer count-runs mechanically but drops acceptance to 42–49 % and lands *below* baseline. Feature injection is a trained contract with a specific target, not an interface. **Quantization is nearly free:** requantizing drafters from Q8_0 to Q4_0 leaves acceptance unchanged (65 vs. 64 %) and *raises* end-to-end speed 4–10 % (the drafter is bandwidth-bound), for both the selector-based v2 and the logit-sampling v1-consistent with the upstream author’s BF16/Q8/Q4 acceptance triplet (4.92/5.08/5.03 tokens/step). Greedy caveat: across drafter quants the accepted *text* can shift at floating-point ties in verify-batch boundaries (295 vs. 297 tokens on one prompt); outputs remain greedy-legal.

E. Where the incumbents hold

Speculation is not one race. Under two concurrent streams the v2 drafter’s aggregate (41–51 tok/s across drafter quants) stays below the MTP head’s multi-stream aggregate (53–61): block drafting is a *solo* weapon on this hardware, though CUDA-class GPUs reportedly scale it well. On the 3.6er, whose native MTP head chains unusually well, MTP depth-2 keeps the code crown (50.3 vs. 42.9 wall). And on the 35B MoE, a trained drafter finally beats n-gram lookup on code (66.4 vs. 59.5 wall, +12 %)-while lookup keeps prose (61.7 vs. 46.1), feeding on repetitive n-grams. Every incumbent survives somewhere.

VI. The Gate

The matrix collapses into a two-input rule, both inputs cheap at serving time (content class from the template/reasoning flag or a running accepted-length statistic; depth from n_{past}):

TABLE III
Deployment gate distilled from §V (this hardware class)

Regime	Policy
code / reasoning, depth < ≈10k	block drafter (v2 if trained for target)
prose, any depth	MTP head; lookup on MoE
depth ≈10–15k	block drafter with n_{max} lowered (≈3)
deeper, or parallel streams	MTP / no speculation
no trained drafter for target	do not transfer one; MTP or lookup

None of this is visible in a single-number benchmark. A serving stack that switches drafter by class and depth captures each regime’s winner; ours currently switches by profile, with the automatic gate as the obvious next step.

VII. Threats to Validity

This is a single-machine, single-model-family study: one GPU architecture (gfx906, memory-bound, modest FP16 throughput), Qwen-family hybrids, greedy decoding, German/English prompts, small *n* per cell (repeats within run, cross-checked across serving stacks). Ratios and mechanisms should transfer better than absolute numbers; the EAGLE-3 verdict in particular is a statement about this cost structure, not about EAGLE-3.

One incident is worth disclosing. During the measurement campaign a second autonomous agent session benched the same GPUs concurrently; the contention masqueraded convincingly as thermal throttling and contaminated two cells before an arbitration protocol (exclusive re-runs under a file lock, cooling gate, first-run discard, *n*=8) separated real effects from interference-and surfaced a genuine artifact in the process (the ≈10 % first-response-after-load penalty). Autonomous benchmarking needs the same hygiene as multi-user clusters: advisory locks (with `flock -o`, lest daemonized children inherit and hold them), kills only by remembered

PID, and suspicion of any effect first observed without exclusivity.

VIII. Related Work

Speculative decoding was introduced for autoregressive drafters [1], [2]; Medusa [5] and MTP heads [3], [4] moved drafting into the target; EAGLE-3 [6] extrapolates target features with a small autoregressive head. DFlash [7] replaces the autoregressive drafter with block diffusion and reports the acceptance lengths we confirm here. Gated DeltaNet [10] and related linear-attention designs supply the recurrent layers whose non-invertible state motivates the RS-ring; to our knowledge, bounded snapshot rollback for speculative verification on such hybrids has not been described in the serving literature, though checkpointing recurrent state is a classic idea in other contexts.

IX. Conclusion

Block-speculative decoding on recurrent-hybrid LLMs needs exactly one new mechanism-bounded, exact rollback of recurrent state-and the RS-ring provides it at a transparent memory price with a hard, assertable edge ($K = n_{\max}$). Once speculation runs, the interesting result is not the headline multiplier but its decomposition: accepted-length differences make reasoning traces the fastest content class, trained-window fidelity decides whether depth is friend or foe, and every drafter family keeps a niche. On two retired 16 GB GPUs the combination lifts a 27B hybrid to 55 tokens/s single-stream-but the durable takeaway is the gate, not the number: *speculate by content class and depth, or you are averaging over regimes your users experience separately.*

Acknowledgment

Experiments, analysis, and manuscript were prepared with substantial AI assistance (Claude, Anthropic) operating the author's workstation under the author's direction; all measurements ran on the author's hardware and are reproducible from the project's benchmark logs (campaigns CR–CW). The port builds on the llama.cpp project [9], the DFlash pull request by its authors, and the z-lab drafter releases.

References

[1] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *Proc. ICML*, 2023.

[2] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," arXiv:2302.01318, 2023.

[3] F. Gloeckle, B. Youbi Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve, "Better & faster large language models via multi-token prediction," in *Proc. ICML*, 2024.

[4] DeepSeek-AI, "DeepSeek-V3 technical report," arXiv:2412.19437, 2024.

[5] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao, "Medusa: Simple LLM inference acceleration framework with multiple decoding heads," in *Proc. ICML*, 2024.

[6] Y. Li, W. Wei, C. Zhang, and H. Zhang, "EAGLE-3: Scaling up inference acceleration of large language models via training-time test," arXiv: 2503.01840, 2025.

[7] J. Chen, Y. Liang, and Z. Liu, "DFlash: Block diffusion for flash speculative decoding," arXiv:2602.06036, 2026.

[8] Qwen Team, Qwen3.6 and Qwen3.8 model releases, Hugging Face model cards, 2026.

[9] G. Gerganov and contributors, "llama.cpp: LLM inference in C/C++," github.com/ggml-org/llama.cpp, 2023–.

[10] S. Yang, J. Kautz, and A. Hatamizadeh, "Gated delta networks: Improving Mamba2 with delta rule," arXiv:2412.06464, 2024.