CATCHUP: REPLAY THE FIRST, REUSE THE REST FOR PREFIX CACHING IN HYBRID LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid language models can incur substantial recovery latency even when a long prefix is cached: token-indexed full-attention (FA) key–value (KV) caches can match beyond the latest recurrent checkpoint, a snapshot of recurrent-layer states at a prefix boundary. Exact recovery recomputes the full model across this gap, limiting the latency benefit of prefix reuse. Denser checkpoints reduce the gap but increase state storage. We introduce CatchUp, which loads the checkpoint and replays a recent tail only through the recurrent layers preceding the first FA layer. Unlike deeper recurrent layers, which require contextual hidden inputs from the preceding network, these layers can be replayed from token embeddings without an additional per-token input cache. A configurable fraction of the checkpoint gap controls replay work. Deeper recurrent layers retain their checkpoint states, while all FA layers reuse matched KV; the full model then processes the uncached suffix. Across three hybrid models on LongBench question answering and summarization, replaying 10% of the checkpoint gap achieves 4.32–7.94$\times$ paired time-to-first-token (TTFT) speedups over full-model recovery (Exact) at an 8K gap on 28 timing inputs. Quality differences from Exact range from -1.83 to $+0.14$ points on all 139 quality questions at that gap. With a 50% window, Qwen3.6-27B reaches 47.39, close to Exact at 47.44. Fixed-history tool-use evaluation on BFCL and long-context evaluation on RULER extend the study to input budgets up to 64K. On RULER at a 64K budget and a 32K gap, CatchUp differs from the full-prefix quality reference (Reference) by -2.61 to $+0.20$ points across the three models, compared with -4.44 to -0.07 for Tail-Replay.

1 INTRODUCTION

Requests over shared documents and multi-turn conversations repeatedly process common context. Prefix caching reuses inference states across requests (Kwon et al., 2023; Zheng et al., 2024; Gim et al., 2024). Hybrid models interleave full-attention (FA) and recurrent layers (Lieber et al., 2024; Qwen Team, 2026a). FA key–value (KV) entries are token-indexed; recurrent checkpoints save all recurrent-layer states at a prefix boundary. When FA KV matches beyond the latest checkpoint, exact recovery recomputes the full model across the gap, limiting the latency benefit of the cached prefix.

A layer group consists of recurrent layers followed by an FA layer; the first is group 0. Figure 1(a) illustrates the recovery bottleneck: the checkpoint is at p, while FA KV matches through q. Full-model recovery (Exact) executes every layer across this gap, including FA computation whose KV is already cached. Thus a long cache hit does not by itself guarantee a low-latency continuation.

Saving checkpoints more densely can shorten the gap, but each snapshot stores the states of all recurrent layers. Admission and placement policies manage this storage tradeoff (Pan et al., 2025; Shirokikh & Nikolenko, 2026). Once a request arrives, recovery must work with the checkpoint that is available. The question is how to reuse the matched FA prefix while limiting the computation and additional storage needed to recover the recurrent states.

Tail-Replay separates FA KV reuse from recurrent recovery (Liu et al., 2026c). In Figure 1(b), it reconstructs every group's recurrent states from zero over a recent prefix tail. For deeper groups, it caches contextual inputs: hidden vectors produced by the preceding network. This replaces full-model recomputation with recurrent replay, but requires additional per-token storage. An existing checkpoint already encodes earlier history and offers a different starting point for recovery.

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

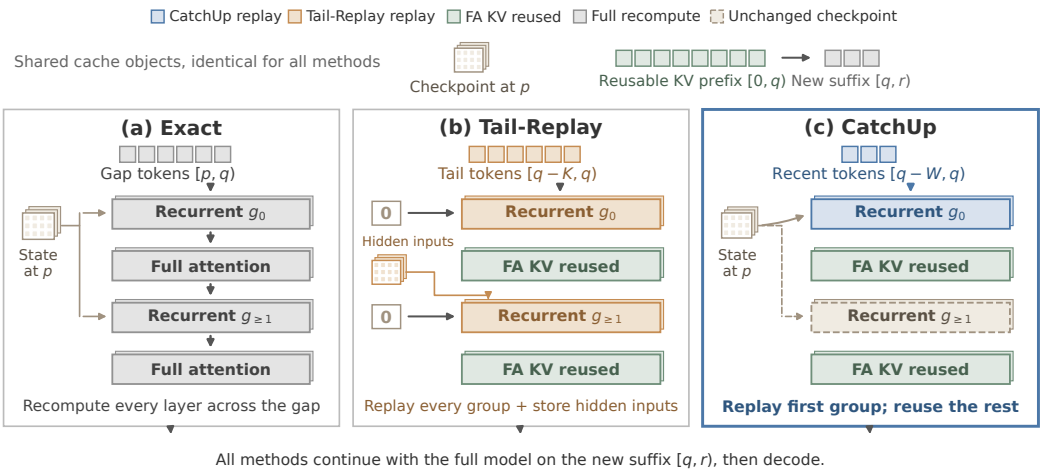

Figure 1: Hybrid-prefix recovery at a cache hit: recurrent checkpoints end at p, while full-attention (FA) KV extends to q. Squares denote token/KV entries; matrix stacks denote saved states or hidden inputs; stacked bars denote layer groups. Exact recomputes every layer over the gap. Tail-Replay zero-starts every recurrent group using cached inputs. CatchUp initializes from the checkpoint and replays only g_0; deeper states retain their values at p. Colors distinguish recomputation, replay, KV reuse, and unchanged states. This is a dataflow schematic, not a runtime plot; Exact has no forced split at q.

Replay-input availability motivates our design. Deeper groups require contextual inputs from the preceding network, including FA layers; supplying them requires retaining hidden vectors or executing that network. The first recurrent group instead starts from token embeddings. CatchUp therefore loads the checkpoint and replays only this group's recurrent layers over a recent tail of the gap, as shown in Figure 1(c). A gap-proportional replay window controls recovery work without an additional contextual-input cache. Deeper recurrent states retain their checkpoint values, all FA layers reuse matched KV, and the full model then processes the uncached suffix normally.

Our contributions are as follows:

- A checkpoint-initialized recovery method that combines token-driven first-group replay with matched FA KV reuse. Its gap-proportional budget controls recovery work without an additional per-token hidden-state cache.
- An analysis of checkpoint and replay-input storage, a cost model relating recovery work to the gap and uncached suffix, and a conditional state-error bound for recurrent replay.
- A three-model evaluation of task quality and recovery latency, identifying configurations where partial replay brings task scores close to full-model recovery at substantially lower latency.

2 Related Work

Prefix reuse in hybrid models. PagedAttention, SGLang, and Prompt Cache manage or reuse attention KV across requests (Kwon et al., 2023; Zheng et al., 2024; Gim et al., 2024). Hybrid models add recurrent state-space or gated linear-attention layers (Gu & Dao, 2024; Dao & Gu, 2024; Yang et al., 2025; Lieber et al., 2024; Qwen Team, 2026a), whose states require checkpoints rather than token-indexed slicing. Hybrid serving engines coordinate these checkpoints with KV reuse.[1]

Checkpoint management and compression. Marconi optimizes checkpoint admission and eviction (Pan et al., 2025); Sparse Prefix Caching optimizes placement (Shirokikh & Nikolenko, 2026). DASC retains long-retention state units and zero-fills or reconstructs omitted units (Yu et al., 2026).

[1]Implementation details: vLLM Hybrid KV Cache Coordinator (commit `2cf0a6915ce5`) and SGLang Unified Radix Cache.

Checkpoint management determines which snapshots are available, while compression changes the state retained within each snapshot. CatchUp starts from an available complete checkpoint and reduces the computation needed to resume beyond it.

State reconstruction and segment reuse. Tail-Replay reconstructs every group's recurrent states from zero using cached contextual inputs (Liu et al., 2026c). CatchUp instead initializes from a checkpoint and replays group 0 from token IDs. Its replay window is a fraction of the checkpoint gap, whereas Tail-Replay's is a fraction of the matched prefix. Table 1 contrasts their storage and recovery paths. LinearKV and HYPIC address position-independent reuse of independently cached segments through state initialization, composition, or attention repair (Liu et al., 2026b;a); CatchUp recovers an identical leading prefix.

Table 1: Features of hybrid-prefix recovery. Checks denote the stated feature, not a quality ranking. All methods use FA KV infrastructure.

Method	Stores checkpoints	Compresses checkpoints	Replays only g_0	Recovery path
vLLM / SGLang	✓	×	×	Full-model resume
Marconi	✓	×	×	Full-model resume
Sparse Prefix Caching	✓	×	×	Full-model gap recomputation
DASC-nr / DASC-wr	✓	✓	×	Zero-fill / zero-start refresh
Tail-Replay	×	×	×	Zero-start all-group replay; cached contextual inputs
CatchUp	✓	×	✓	Checkpoint-initialized replay; token IDs

3 Background and Problem

Let $x_{0:r}$ be a prompt; intervals are half-open and position t denotes the boundary after $x_{0:t}$. A *group* comprises consecutive recurrent layers followed by an FA layer, indexed $g = 0, \ldots, G-1$. The recurrent portion $\mathcal{R}_g$ carries matrix states S, which accumulate sequence history, and short-convolution buffers C, which retain recent projected inputs. We collect both into $Z_{g,t}$, the group state before processing token t:

$$Z_{g,t} = \{S_t^\ell, C_t^\ell\}_{\ell \in \mathcal{R}_g}, \qquad Z_{g,t+1} = F_g(Z_{g,t}, \mathbf{u}_{g,t}; t). \tag{1}$$

The update maps the current state and group input $\mathbf{u}_{g,t}$ to the next state, and F_g includes the intervening normalization, residual, and feed-forward operations but excludes the following FA layer. Qwen3.5-4B has eight groups, each with three Gated DeltaNet layers and one FA layer (Qwen Team, 2026a). With convolution kernel width $\kappa = 4$, its buffer stores $\kappa - 1 = 3$ projected inputs; position and other continuation metadata are maintained separately.

Let q be the farthest reusable FA-prefix boundary and p the latest boundary with both reusable KV and a complete recurrent checkpoint. Formally, $\mathcal{H}_{\mathrm{FA}}$ contains FA-prefix boundaries at which the retained KV supports valid continuation and $\mathcal{P}$ contains complete recurrent-checkpoint positions, both including the initial boundary zero. Define

$$q = \max \mathcal{H}_{\mathrm{FA}}, \qquad p = \max(\mathcal{H}_{\mathrm{FA}} \cap \mathcal{P}), \qquad D = q - p. \tag{2}$$

Exact resumes every layer at p and processes $[p, r)$. The recovery problem is to reuse FA KV through q while approximating the missing recurrent states there. The checkpoint gap D, total input length r, and uncached suffix length $L = r - q$ are distinct quantities.

Replay-input availability. For the first group, $\mathbf{u}_{0,t} = \mathrm{Embed}(x_t)$, with any deterministic positional preprocessing. Deeper inputs depend on the preceding network, including FA layers; cached KV does not directly provide these hidden inputs. Replaying deeper groups requires storing them or executing that network. This motivates spending a bounded recovery budget on the first group ($g = 0$) while retaining the deeper checkpoint states.

4 CatchUp Design

4.1 Checkpoint-Initialized First-Group Replay

At a cache hit, acquire FA KV through q and the complete checkpoint at p. For $D > 0$, choose a replay fraction $0 < \rho \leq 1$, a window $W = \lceil \rho D \rceil$, and its starting position $s = q - W$; for $D = 0$, skip replay. Figure 2 shows group-0 replay over $[s, q)$ followed by full-model suffix execution.

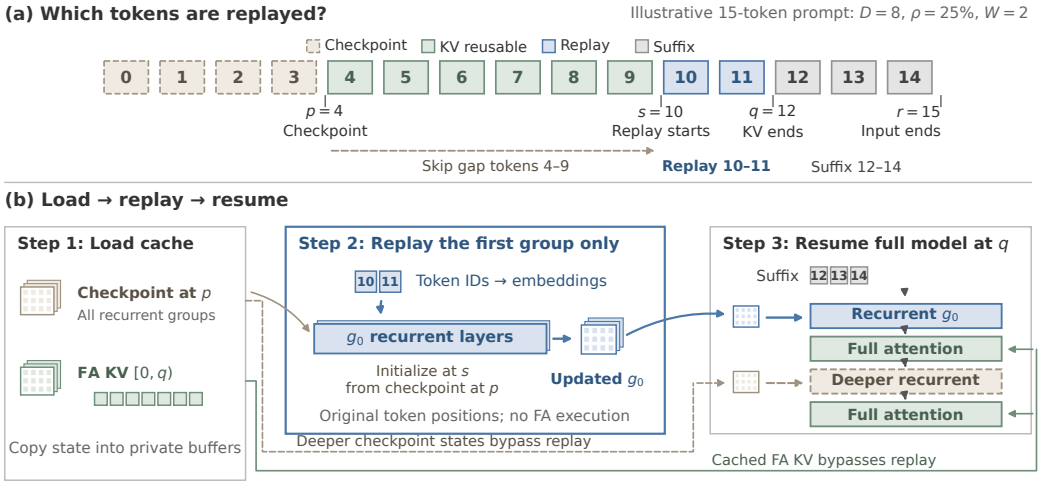

Figure 2: CatchUp through a concrete token example and its recovery dataflow. (a) Each square is one token; p, s, q, r label boundaries between tokens, not elapsed time. With $p = 4$, $q = 12$, and $\rho = 25\%$, $D = 8$ and $W = 2$: tokens 4–9 are skipped, 10–11 are replayed, and 12–14 form the suffix. This toy example is not an experimental setting. (b) Checkpoint state initializes first-group replay from token embeddings. Deeper checkpoint states and FA KV bypass replay and join the updated first-group state at q; the full model then processes the suffix. Recovery remains approximate.

Let $\widehat{Z}$ denote the approximate recovery state. Copy checkpoint tensors into private, writable state for this request and execute

$$\begin{aligned} \widehat{Z}_{0,s} &\leftarrow Z_{0,p}, \\ \widehat{Z}_{0,t+1} &= F_0(\widehat{Z}_{0,t}, \mathrm{Embed}(x_t); t), \quad s \leq t < q, \\ \widehat{Z}_{g,q} &\leftarrow Z_{g,p}, \quad g > 0. \end{aligned} \tag{3}$$

The assignment initializes replay at s with the old state at p, skipping $[p, s)$. Replay uses original token positions and executes only group 0, including its intervening operations. With continuation metadata set to q, all layers process $x_{q:r}$ and decode normally. Published FA KV and checkpoints remain immutable.

4.2 Storage and Computational Cost

Let M_{ckpt} be the full recurrent-checkpoint bytes, F FA KV bytes per token, and H contextual-input bytes per token for deeper replay boundaries. For a cache holding Q tokens in total (distinct from the request boundary q) and checkpoint spacing Δ dividing Q,

$$M_{CatchUp} = QF + (Q/\Delta)M_{\mathrm{ckpt}}, \qquad M_{\mathrm{Tail}} = Q(F + H). \tag{4}$$

The shared KV term cancels: CatchUp uses less persistent tensor storage when $\Delta > M_{\mathrm{ckpt}}/H$. This tensor-capacity model counts positive-grid checkpoints, excluding weights and runtime overhead. Checkpoint spacing Δ differs from the request gap D. Section 5.3 evaluates these costs.

Let c_0 be first-group replay cost per token, τ separate-pass overhead, and $T_{\mathrm{pf}}(a, r)$ native full-model prefill time over $[a, r)$ after cache installation. With Δ_{io} the additional transfer/installation cost

relative to Exact,

$$T_{\text{Exact}} - T_{CatchUp} \approx \underbrace{T_{\text{pf}}(p, r) - T_{\text{pf}}(q, r)}_{\text{full-model computation saved}} - (Wc_0 + \tau) - \Delta_{\text{io}}. \tag{5}$$

The saving subtracts replay and transfer costs from bypassed computation; Exact runs $[p, r)$ without a forced split at q. A larger gap can increase bypassed computation, while a longer suffix adds shared work. Tail-Replay uses a matched-prefix fraction η, replaying $K = \lceil \eta q \rceil$ tokens in every recurrent group from zero.

4.3 Conditional State-Error Analysis

For one Gated DeltaNet head, let S_t be the reference matrix state and $\widehat{S}_t$ its replay counterpart. The update is $S_{t+1} = \mathcal{A}_t(S_t) + B_t$, where $\mathcal{A}_t(S) = \alpha_t S(I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)$ and $B_t = \beta_t \mathbf{v}_t \mathbf{k}_t^\top$. Here α_t is the forget gate, β_t the write strength, and $\mathbf{k}_t, \mathbf{v}_t$ the key and value. For shared gates, keys, and values, the new writes cancel: $E_t = \widehat{S}_t - S_t$ obeys $E_{t+1} = \mathcal{A}_t(E_t)$.

Proposition 1 (Replay error)**.** *Suppose replay and reference share $\alpha_t, \beta_t, \mathbf{k}_t, \mathbf{v}_t$ on $[a, q)$, with $\alpha_t \in (0, 1)$, $\beta_t \in [0, 2]$, and $\|\mathbf{k}_t\|_2 = 1$. Defining $\Phi_{a,q} = \mathcal{A}_{q-1} \circ \cdots \circ \mathcal{A}_a$, we have*

$$E_q = \Phi_{a,q}(E_a), \qquad \|E_q\|_F \le \Big(\prod_{t=a}^{q-1} \alpha_t\Big) \|E_a\|_F. \tag{6}$$

Here $\|\cdot\|_F$ is the Frobenius norm. Since $\|I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\|_2 \le 1$, each shared update contracts the error by at most α_t.

In the first layer, $\kappa - 1$ replay tokens replace the loaded convolution history. For $W \ge \kappa - 1$, apply the bound from $a = s + \kappa - 1$ with the resulting error E_a; use $a = s$ if histories already agree. Later layers receive perturbed upstream outputs, so this alignment alone does not ensure shared transitions there. Changing W changes both the starting error and the transition product; task-quality effects are measured in the experiments.

Interpreting the state bound. For shared transitions on $[s, q)$, zero-start and checkpoint-start errors are $-\Phi_{s,q}(S_s)$ and $\Phi_{s,q}(S_p - S_s)$, respectively. The contraction bound applies to either mismatch and does not establish which is smaller. CatchUp replays only group 0: deeper recurrent states remain at p, omitting their updates over $[p, q)$, so the full state at q remains approximate even when $W = D$ makes group 0 exact. Tail-Replay also differs in replayed groups and spans; Appendix A gives the derivation.

5 Evaluation

We evaluate CatchUp on Qwen3.5-4B, OLMo-Hybrid-7B (Merrill et al., 2026), and Qwen3.6-27B (Qwen Team, 2026b), examining recovery latency, persistent-cache capacity, and task quality in that order. LongBench supplies the main comparison; BFCL and RULER extend it to tool use and longer inputs.

5.1 Experimental Setup

Inputs and recovery policies. The fixed LongBench subset (Bai et al., 2024) contains 139 questions from 120 contexts: NarrativeQA, HotpotQA, MuSiQue, and QMSum contribute 36/36/40/27 questions. A fixed subset of 28 questions from 26 contexts (7/6/10/5 per task) supplies timing measurements. Every input supports $D \in \{1, 2, 3, 4, 8\}$K in all models; 1K=1,024 tokens. Under a 16,384-token input budget, we place q near the input end, at the last 1,024-token boundary leaving at least 128 suffix tokens. The checkpoint is D tokens before q; input and suffix stay fixed across gaps.

A producer builds FA KV and complete checkpoints before recovery. Exact resumes native full-model execution from p over $[p, r)$. CatchUp reuses FA KV through q and replays $W = \lceil \rho D \rceil$ tokens in group 0. Our native-vLLM implementation of Tail-Replay (Liu et al., 2026c) uses the same inputs, generation settings, and scoring as the other methods. It zero-starts every recurrent group over

$K = \lceil \eta q \rceil$ tokens using cached contextual inputs. The measured Tail-Replay configurations retain each replayed group's terminal feed-forward block. The common comparison uses $\rho = \eta = 10\%$, with different denominators D and q; budget sweeps cover $\rho \in \{5, 10, 50, 100\}\%$ and $\eta \in \{5, 10\}\%$. The no-replay ablation is reported in Appendix D.

Timing and scoring. Native-vLLM runs use A100 80GB GPUs, BF16, single requests, eager execution, and tensor parallelism one. FA KV is GPU-resident; checkpoints and Tail-Replay inputs are in CPU pinned memory. Thus the original Tail-Replay optimization overlapping host-to-device FA KV transfer with replay is not used. Synchronized time to first token (TTFT) spans pretokenized entry through the in-process return of `generate(max_tokens=1)`, including state transfer, private-state initialization, replay, suffix processing, and sampling. Cache construction, model loading, compilation, and audit probes are outside this interval. Each configuration has three warmups and five measurements; we use per-input medians. Latency means weight tasks equally. Speedups average paired per-input Exact/method ratios within each task and then across tasks.

Quality uses greedy generation with output limits of 128/32/32/512 tokens in the task order above, retaining capped outputs. Both Qwen models run with thinking disabled (`enable_thinking=False`) under each model's official chat template with a single user message; BFCL likewise applies each model's official chat template to the frozen messages and tools. QA uses token F1 and QMSum uses ROUGE-L; macro scores on a 0–100 scale weight tasks equally. LongBench intervals use 20,000 paired bootstrap resamples of whole contexts within tasks. They are pointwise, without adjustment for configuration selection. Quality results use all 139 questions; joint quality–latency results use the same 28 timing questions. Exact follows its native prefill schedule at each gap, whose partitioning can affect greedy outputs.

5.2 Recovery Latency

Figure 3 compares recovery time as the checkpoint gap grows. Exact executes the full model across the gap, whereas CatchUp spends its recovery work on a short first-group replay before the shared suffix.

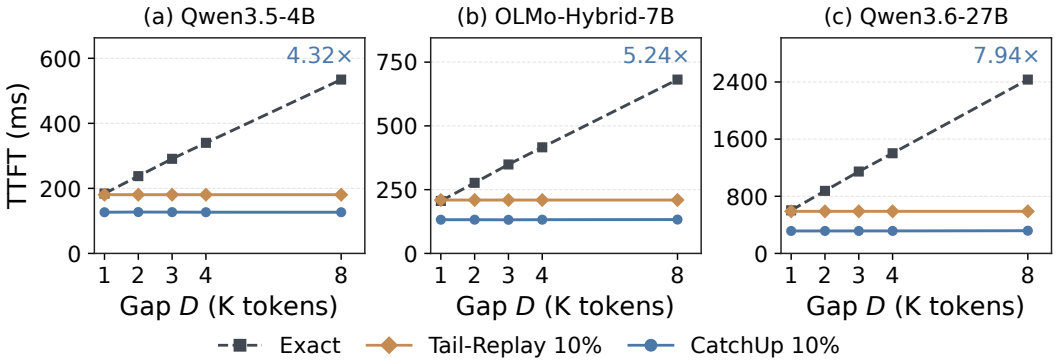

Figure 3: Recovery TTFT on the same 28 LongBench timing questions. Each model panel shows all five gaps; lower is faster. Points are task-balanced means of per-input median TTFTs. CatchUp and Tail-Replay use 10% of D and q, respectively. Blue annotations report paired Exact/CatchUp speedups at 8K, rather than ratios of the plotted means.

For Qwen3.5-4B, OLMo-Hybrid-7B, and Qwen3.6-27B at 8K, CatchUp takes 126.6/132.7/317.9 ms, compared with Exact's 534.6/681.6/2,433.7 ms. Paired speedups reach 4.32/5.24/7.94×. Tail-Replay takes 180.5/209.4/589.2 ms at its 10% setting. Thus CatchUp reduces recovery time relative to both full-model recomputation and the evaluated Tail-Replay implementation. Its mean TTFT changes little across the tested gaps, while Exact's grows with the recomputed span. A larger replay budget remains inexpensive relative to Exact: Qwen3.6-27B at 8K with 50% replay takes 346.3 ms and achieves 7.32× paired speedup.

5.3 Persistent Cache Capacity

The latency benefit requires checkpoints but no additional per-token contextual-input cache. Figure 4 evaluates Equation (4), including the shared FA KV and all periodic checkpoints in a reusable prefix cache.

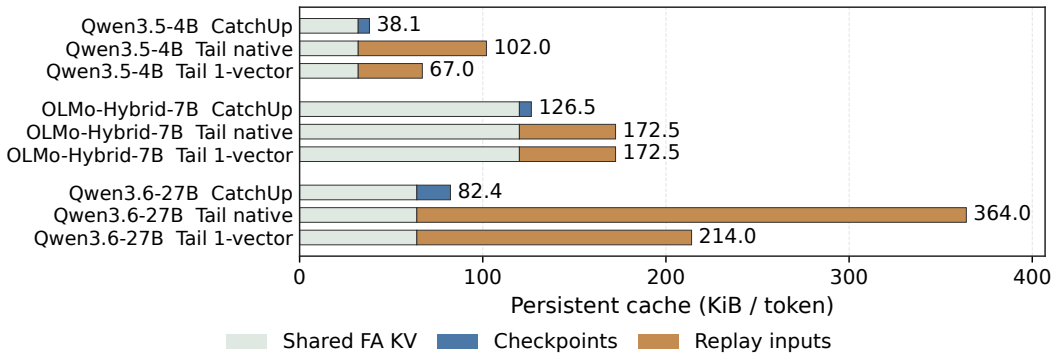

Figure 4: Amortized persistent capacity per cached token at spacing $\Delta = 8$K in a 24K prefix (three checkpoints). One complete checkpoint occupies 49.1/52.2/146.8 MiB for Qwen3.5-4B/OLMo-Hybrid-7B/Qwen3.6-27B; its amortized cost is 6.14/6.53/18.35 KiB per token. Bars add this cost or Tail-Replay inputs to shared FA KV. Native Qwen inputs store hidden and residual vectors; the one-vector variant stores one per deeper boundary. OLMo-Hybrid-7B uses one in both variants.

At 8K spacing, CatchUp uses 37.4/73.3/22.6% of native Tail-Replay's capacity for Qwen3.5-4B/OLMo-Hybrid-7B/Qwen3.6-27B; the one-vector ratios are 56.9/73.3/38.5%. Wider spacing amortizes checkpoints over more tokens. All three models use less capacity at spacings of 2K or more; Qwen3.5-4B crosses the one-vector baseline at about 1.40K. These estimates count persistent tensors across the cached prefix, excluding weights and runtime overhead. Request timing includes only the required transfers. Appendix B.1 gives the tensor accounting and Figure 7 shows the spacing sweep.

5.4 Task Quality and Replay Budget

We next examine task quality relative to full-model recovery (Exact), across replay budgets at the largest evaluated checkpoint gap. Figure 5 shows every tested budget at the largest LongBench gap, $D = 8$K.

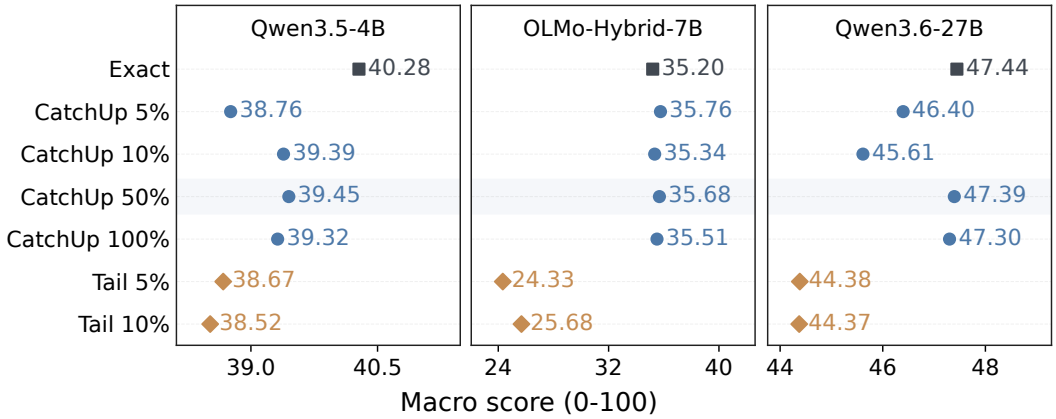

Figure 5: Task quality at $D = 8$K on all 139 LongBench questions. All three models and tested CatchUp and Tail-Replay budgets are shown alongside Exact. Labels give absolute macro scores; horizontal scales differ by model. The shaded row marks 50% replay.

Quality at the common budget. At 10%, CatchUp scores 39.39/35.34/45.61 on Qwen3.5-4B/OLMo-Hybrid-7B/Qwen3.6-27B, compared with Exact at 40.28/35.20/47.44. Differences range from -1.83 to $+0.14$ points. At the same setting, Tail-Replay scores 38.52/25.68/44.37; CatchUp is higher by 0.87/9.66/1.24 points. On OLMo-Hybrid-7B, Tail-Replay more frequently continues generating after its initial answer. A sensitivity check takes the first nonempty line for all methods on HotpotQA and MuSiQue, leaving the other tasks unchanged. The four-task macro gaps decrease from 9.52 to 1.66 points for Exact–Tail-Replay and from 9.66 to 4.25 for CatchUp–Tail-Replay; main results retain whole-output scoring (Appendix D.4). Quality uses all 139 questions; timing uses the fixed 28-question subset.

Replay budget at a large gap. At $D = 8$K, Qwen3.6-27B with 50% replay reaches 47.39, versus Exact at 47.44 and 10% replay at 45.61. Increasing the budget to 100% gives 47.30. This configuration brings the score close to full-model recovery by replaying half of the gap in group 0. Appendix D reports the complete gap sweep and no-replay ablation.

5.5 Tool-Use Recovery

We evaluate second-turn recovery on the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025). We freeze 40 independent scenarios and a first-turn history for each before comparing methods. The main comparison uses the same 20 long-history scenarios supporting every $D \in \{1, 2, 3, 4\}$K in both Qwen models, with identical history text and reset tool environments. Both replay methods use 10% of their respective denominators. Greedy generation allows 2,048 tokens per output, up to 20 tool calls and 21 outputs, within a 32,768-token context. Success requires both official state and response checks.

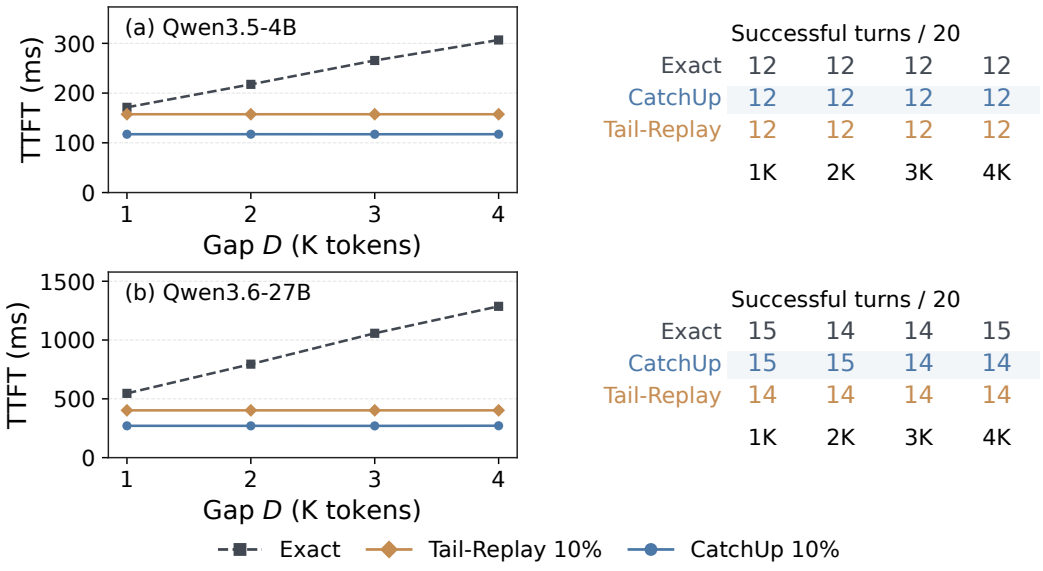

Figure 6: Second-turn recovery on the same 20 BFCL scenarios. Panels (a) and (b) show Qwen3.5-4B and Qwen3.6-27B. Each row shows mean per-scenario median TTFT on the left and successful-turn counts on the right for Exact, CatchUp, and Tail-Replay. Replay fractions are 10% of D for CatchUp and q for Tail-Replay.

Figure 6 pairs recovery time with task success. At 4K, CatchUp reduces mean TTFT from Exact's 306.8 to 117.3 ms on Qwen3.5-4B and from 1,286.5 to 271.3 ms on Qwen3.6-27B; Tail-Replay takes 157.4/402.2 ms. Geometric means of paired Exact/CatchUp ratios are 2.62× and 4.82×, with 95% intervals $[2.575, 2.663]$ and $[4.515, 5.112]$ from 10,000 paired scenario resamples. TTFT ends at the first assistant token, before tool execution.

The right panels show comparable successful-turn counts. At 4K, Exact/CatchUp/Tail-Replay succeed on 12/12/12 scenarios for Qwen3.5-4B and 15/14/14 for Qwen3.6-27B. Across gaps, CatchUp differs from Exact by at most one successful scenario in this common cohort; all corresponding paired intervals include zero. Appendix E reports the complete intervals, the no-replay ablation, and larger cohorts available at shorter gaps.

5.6 Long-Input Quality up to 64K

We extend quality evaluation to five RULER tasks (Hsieh et al., 2024): single-needle retrieval, multi-key retrieval, variable tracing, frequent-word extraction, and QA. Each model and 16K/32K/64K input budget uses 100 inputs per task. Gaps within a length use identical inputs; cohorts differ across lengths. These quality measurements use Hugging Face models with FLA recurrent kernels (Yang & Zhang, 2024). Reference directly loads the producer state at q and serves as the full-prefix quality baseline. Exact in the native-vLLM latency experiments instead resumes full-model execution from p. Both replay methods use 10%, with shared FA KV, inputs, and generation limits across methods. Generation allows up to 128/128/30/50/32 new tokens for the five tasks in the order above.

At the 64K input budget and 32K gap, CatchUp scores 91.60/83.36/94.33 on Qwen3.5-4B/OLMo-Hybrid-7B/Qwen3.6-27B, compared with Reference at 91.40/85.97/94.80 and Tail-Replay at 91.33/81.53/94.60. On OLMo-Hybrid-7B, CatchUp exceeds Tail-Replay by 1.83 points; its deficit from Reference is 2.61 points, compared with Tail-Replay's 4.44. Several Qwen retrieval tasks are at or near full scores, leaving limited room for score changes. Across all 27 settings, CatchUp's macro scores differ from Reference by -2.61 to $+0.56$ points and from Tail-Replay by -1.20 to $+4.58$ points, with the relative ranking varying by configuration. Because length cohorts differ, this sweep compares methods within each input budget rather than tracking the same inputs as context grows. Appendix F reports five-task macro scores and paired no-replay comparisons for all 27 model–length–gap settings.

6 Conclusion

CatchUp combines checkpoint-initialized first-group replay with matched FA KV reuse, avoiding an additional contextual-input cache. At an 8K gap, 10% replay achieves 4.32–7.94$\times$ paired TTFT speedups over full-model recovery across three models. At 8K checkpoint spacing, its persistent tensor capacity is 22.6–73.3% of native Tail-Replay's. On LongBench, the common 10% budget changes scores by -1.83 to $+0.14$ points relative to Exact; Qwen3.6-27B with 50% replay reaches 47.39 versus Exact's 47.44. BFCL and RULER extend the evaluation to tool use and input budgets up to 64K. Within RULER, relative quality rankings vary across model–length–gap settings. CatchUp offers a configurable tradeoff between recovery work, cache storage, and task quality.

Checkpoint spacing and replay fraction provide two distinct controls. Spacing determines how much recurrent state is retained across the cached prefix, whereas the replay fraction determines how much of a request's checkpoint gap is revisited. Their effects meet at recovery: wider spacing amortizes checkpoint storage, while the selected replay window sets the first-group computation before suffix processing. The budget sweep shows why this choice should be evaluated alongside task quality rather than set solely from a latency target. Larger windows can bring a configuration closer to full-model recovery, but the effect varies with the model and task. The method's structural advantage is that this adjustment uses token embeddings and existing checkpoint tensors, without storing deeper contextual inputs at every cached token. Matched FA KV preserves access to the prefix during subsequent full-model computation. The latency model links these choices to the uncached suffix: as shared suffix computation grows, it contributes more to both recovery paths. Checkpoint management determines which starting states are available, and CatchUp determines how to resume from the selected state. This separation connects the method to existing prefix-cache policies while keeping the recovery budget explicit. Together, these results characterize first-group replay as a practical recovery option for hybrid prefix caches, with storage, request latency, and quality assessed as separate, measurable dimensions.

Reproducibility Statement

Sections 4 and 5 specify recovery, inputs, baselines, scoring, timing, and primary recovery comparisons. Supplementary sections provide the complete LongBench quality grid, the proof, tensor accounting, the implementation contract, LongBench per-task results at the common 10% budget, paired intervals for the specified comparisons, and additional diagnostics.

AI Use Statement

In this work, we used generative AI tools to assist with translation, to refine the research hypothesis and the experimental protocol, to implement the method and evaluation code, to interpret results, and to formulate mathematical claims and to draft and check the mathematical analysis and its proof. Generating synthetic data sets, cleaning or reformatting datasets, and qualitative or thematic data analysis are not applicable to this work, and we have not used generative AI tools for any other tasks with required disclosure. Additionally, we used generative AI tools to draft parts of the manuscript, to search for and summarize relevant literature, and to create scientific figures. We have reviewed all AI-assisted work: AI-assisted drafts and translations were checked and revised by the authors, AI-assisted code was verified and tested for correctness, and the mathematical claims, analysis, and proof were verified step by step and corrected by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2405.21060.

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 (MLSys)*, 2024. URL https://arxiv.org/abs/2311.04934.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2312.00752.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.06654.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. 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 (SOSP)*, 2023. URL https://arxiv.org/abs/2309.06180.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-Mamba language model. arXiv preprint arXiv:2403.19887, 2024. URL https://arxiv.org/abs/2403.19887.

Yifei Liu, Juntong Wu, Yang Liu, Junhao Hu, Minghao Li, Xiaoxu Chen, and Weihang Chen. HYPIC: Accelerating hybrid-attention LLM serving with position-independent caching. arXiv preprint arXiv:2607.01299v2, 2026a. URL https://arxiv.org/abs/2607.01299v2.

Yirui Liu, Ruoling Qi, Longwen Wang, Xuaner Wu, Jian Chen, Yuxin Jin, Jiawei Shao, and Xuelong Li. LinearKV: One cached state suffices for position-independent caching in hybrid LLMs. arXiv preprint arXiv:2608.11231v1, 2026b. URL https://arxiv.org/abs/2608.11231v1.

Yirui Liu, Ruoling Qi, Xuaner Wu, Penghang Liu, and Jian Chen. Tail-Replay: Escaping the curse of linear attention in prefix caching for hybrid LLMs. arXiv preprint arXiv:2608.30310v1, 2026c. URL https://arxiv.org/abs/2608.30310v1.

William Merrill, Yanhong Li, Tyler Romero, et al. Olmo hybrid: From theory to practice and back. arXiv preprint arXiv:2604.03444, 2026. URL https://arxiv.org/abs/2604.03444.

Rui Pan, Zhuang Wang, Zhen Jia, Can Karakus, Luca Zancato, Tri Dao, Yida Wang, and Ravi Netravali. Marconi: Prefix caching for the era of hybrid LLMs. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025. URL https://arxiv.org/abs/2411.19379.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Mikhail Shirokikh and Sergey Nikolenko. Sparse prefix caching for hybrid and recurrent LLM serving. arXiv preprint arXiv:2605.05219, 2026. URL https://arxiv.org/abs/2605.05219.

Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention mechanism. https://github.com/fla-org/flash-linear-attention, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2412.06464.

Yanqi Yu, Pingwei Sun, Jianchao Tan, Tao Zhang, Yuchen Xie, Xunliang Cai, and Yao Liu. DASC: Decay-aware state compression for hybrid linear-attention serving. arXiv preprint arXiv:2608.30386v1, 2026. URL https://arxiv.org/abs/2608.30386v1.

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 (NeurIPS)*, 2024. URL https://arxiv.org/abs/2312.07104.

A Recurrent State Analysis

A.1 Gated DeltaNet recurrence

For one head of a Gated DeltaNet layer, let $S_t \in \mathbb{R}^{d_v \times d_k}$, $\mathbf{k}_t \in \mathbb{R}^{d_k}$, and $\mathbf{v}_t \in \mathbb{R}^{d_v}$. Its state update is

$$S_{t+1} = \alpha_t S_t\big(I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\big) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top, \tag{7}$$

Here $\alpha_t \in (0,1)$ is a forget gate, β_t a write strength, $\mathbf{k}_t$ an L_2-normalized key, and $\mathbf{v}_t$ a value (Yang et al., 2025). The write strength lies in $(0,1)$ for Qwen3.5 and $(0,2)$ for the evaluated OLMo-Hybrid configuration, which allows negative transition eigenvalues.[2] The query, key, and value pass through a short convolution of kernel size κ, whose buffer C holds the projected inputs of the previous $\kappa - 1$ tokens; $\kappa = 4$ in Qwen3.5.

A.2 Replay initializations

Under shared gates, keys, and values throughout the tail (including its convolution boundary), take $s = q - W$. The following idealized errors compare initializations over the same window W:

$$\widehat{S}_q - S_q = \begin{cases} S_p - S_q & \text{checkpoint without replay,} \\ -\Phi_{q-W,q}\big(S_{q-W}\big) & \text{zero-start replay,} \\ \Phi_{q-W,q}\big(S_p - S_{q-W}\big) & \text{warm-start replay (CatchUp).} \end{cases} \tag{8}$$

The warm-start bound contracts the mismatch $S_p - S_{q-W}$, whereas the zero-start bound contracts S_{q-W}. The skipped interval determines this initial mismatch; the replay transition determines how it propagated. Changing W changes both the starting mismatch and the transition product. In later recurrent layers, changed preceding-layer outputs also perturb gates, keys, and values. Section 5 evaluates the combined effect on task quality and latency.

A.3 Lag of the deeper groups

For one head of a recurrent layer in a deeper group, evaluated along the reference trajectory, the decomposition gives

$$S_q = \underbrace{\Phi_{p,q}(S_p)}_{\text{propagated checkpoint state}} + \underbrace{\sum_{t=p}^{q-1} \Phi_{t+1,q}(B_t)}_{\text{gap updates}}. \tag{9}$$

Installing S_p at q preserves the checkpoint tensor but omits both gap contributions and their transformations of earlier history. The prefix tokens remain accessible through the retained FA KV during suffix processing. Subsequent suffix computation updates all groups from these mixed-age states. With $W = D$, replay follows the complete $[p, q)$ trajectory of g_0 from its checkpoint, recovering that group's reference state in exact arithmetic while retaining the deeper-group lag.

A.4 Proof of Proposition 1

Proof. At every $t \in [a, q)$, both runs use the same α_t, β_t, $\mathbf{k}_t$, and $\mathbf{v}_t$, hence the same linear map $\mathcal{A}_t$ and the same offset B_t. Subtracting the reference update from the replayed one cancels B_t:

$$\widehat{S}_{t+1} - S_{t+1} = \mathcal{A}_t(\widehat{S}_t) - \mathcal{A}_t(S_t) = \mathcal{A}_t\big(\widehat{S}_t - S_t\big). \tag{10}$$

Induction over t from a to q yields $\widehat{S}_q - S_q = \Phi_{a,q}(\widehat{S}_a - S_a)$. For the bound, $\|XY\|_F \le \|X\|_F \|Y\|_2$ gives

$$\|\mathcal{A}_t(X)\|_F = \alpha_t \big\|X(I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\big\|_F \le \alpha_t \|X\|_F \, \big\|I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\big\|_2. \tag{11}$$

Because $\|\mathbf{k}_t\|_2 = 1$, the symmetric matrix $I - \beta_t \mathbf{k}_t \mathbf{k}_t^\top$ has eigenvalue $1 - \beta_t$ along $\mathbf{k}_t$ and eigenvalue 1 on the orthogonal complement, so $|1 - \beta_t| \le 1$ and its spectral norm is at most 1 for $\beta_t \in [0, 2]$. Hence $\|\mathcal{A}_t(X)\|_F \le \alpha_t \|X\|_F$, and composing the maps over $[a, q)$ gives the bound. The argument extends to any later position at which the layer still computes the reference gates, key, and value. It assumes exact arithmetic. □

[2] OLMo Hybrid configuration, Transformers v5.3.0.

For a Mamba-2 layer, $\mathcal{A}_t$ multiplies each head of the state by a scalar decay $a_t \in (0, 1)$ (Dao & Gu, 2024), and the proposition holds with a_t in place of α_t.

B PERSISTENT CACHE STORAGE ACCOUNTING

The storage comparison uses a 24K-token retained prefix and positive checkpoint-grid positions spaced by $\Delta \in \{1, 2, 3, 4, 8\}$K. Thus the numbers of complete checkpoints are 24, 12, 8, 6, and 3. The same FA KV is counted for both methods. Tail-Replay retains contextual inputs across the cached positions; each request fetches its selected tail. This analytical cache-policy comparison is separate from the fixed-hit timing experiment, which prepares only the required replay-input slices.

Table 2: CatchUp checkpoint storage amortized over cached tokens (M_{ckpt}/Δ, KiB per token). Add the model-specific F from Table 3 to obtain the full persistent cache cost. Tail-Replay instead adds its H to the same F.

Spacing Δ	Qwen3.5-4B	OLMo-Hybrid-7B	Qwen3.6-27B
1K	49.125	52.207	146.812
2K	24.562	26.104	73.406
3K	16.375	17.402	48.938
4K	12.281	13.052	36.703
8K	6.141	6.526	18.352

All five spacings reduce persistent cache bytes relative to the native Tail-Replay layouts. At 1K, OLMo-Hybrid-7B has only a small difference: its checkpoint contribution is 52.207 versus 52.5 KiB per token for contextual inputs. With one-vector inputs, the Qwen3.5-4B checkpoint contribution at 1K is 49.125 versus 35 KiB; its break-even spacing is approximately 1.40K. At 2K and larger spacings all three models are below both input representations.

Qwen3.5-4B tensor example. Qwen3.5-4B has 24 recurrent layers, each with 524,288 matrix-state values and 24,576 convolution-history values. FP32 matrices and BF16 history therefore occupy $24(524{,}288 \times 4 + 24{,}576 \times 2) = 51{,}511{,}296$ bytes (49.125 MiB). Its eight FA layers store $8 \times 2 \times 4 \times 256$ BF16 values per token, or 32 KiB. This counts minimal three-token convolution history and excludes allocator padding, metadata, speculative buffers, and runtime duplication.[3]

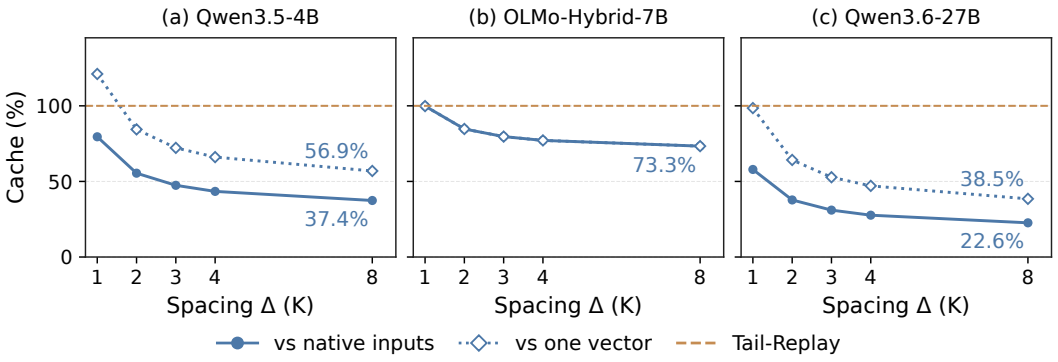

Figure 7: Persistent capacity across checkpoint spacings, normalized to native or one-vector Tail-Replay inputs. The 100% line denotes equal capacity. All five spacings and all three models are shown; the 8K endpoint is decomposed in Figure 4.

[3]Tensor dimensions: Qwen3.5-4B configuration (revision `c7429d5a8ed5`); state layouts: vLLM Mamba state-shape utilities.

B.1 Model tensor payloads

Table 3: Persistent tensor payloads. Checkpoints use FP32 recurrent matrices and BF16 convolution history; FA KV and contextual inputs use BF16. M_{ckpt} is MiB per complete checkpoint; F and H are KiB per cached token.

Model	M_{ckpt}	FA KV F	H (one vector)	H (native)
Qwen3.5-4B	49.125	32	35	70
OLMo-Hybrid-7B	52.207	120	52.5	52.5
Qwen3.6-27B	146.812	64	150	300

C Paged-Cache Integration

A checkpoint supplies S, C, and continuation metadata for every recurrent layer at a consistent token boundary. Cache identity and state-layout requirements are listed in Section C.1. A paged-cache integration exposes both q and p to recovery. Table 4 specifies the interface and object-lifetime contract; Section 5.1 describes the producer-generated caches used in our evaluation.

Table 4: Paged-cache integration contract for CatchUp.

1	Look up legal FA prefix reuse ending at q and select the latest common reusable checkpoint, $p = \max(\mathcal{H}_{\mathrm{FA}} \cap \mathcal{P})$ as in Equation (2), with $D = q - p$ and $W = \lceil \rho D \rceil$ for the configured $0 < \rho \leq 1$.
2	Acquire the FA blocks of $[0, q)$ and the checkpoint at p, validate their identity and availability, and reserve private recurrent state.
3	Copy $Z_{g,p}$ into the private state of every recurrent group g.
4	If $W > 0$, replay the recurrent portion of g_0 on $x_{q-W:q}$ from its copied state, using the original token positions.
5	Set continuation metadata to q, run $x_{q:r}$ through the entire model, and decode. Release acquired references through the engine's normal lifecycle.

Replay includes the operations between the first group's recurrent layers. A terminal feed-forward block may be omitted when its output is discarded. This is an optional optimization; the measured Tail-Replay configurations retain this block.

Under this contract, acquisition pins the FA blocks and checkpoint until their reads finish. Recovery updates private recurrent state while cached FA blocks and checkpoints remain immutable. In the evaluated setting, FA KV stays on the GPU and checkpoint transfer is included in TTFT.

C.1 Recovery Metadata

The following metadata supports the paged-cache integration contract in Table 4. The evaluated requests use checkpoints at prescribed positions, as specified in Section 5.1.

Table 5: Objects and conditions required by an executable recovery plan.

Object	Required information
FA prefix	Prefix and conditioning identity, legal end position q, all required layers, tensor layout, and acquired object generations.
Checkpoint	Position p, the recurrent and convolution states of every recurrent layer, model and adapter configuration, and logical metadata.
Recovery entry	Checkpoint position p, target position q, replay range $[q - W, q)$ for g_0, and private output state for every group.

The integration contract keeps published FA blocks and checkpoints immutable; recovery and decoding use private state or copy-on-write storage. Prefix identity identifies candidate content, while payload availability and object generations establish that an acquired object remains usable. Recovery reads the matched FA prefix and the selected checkpoint, requiring no additional persistent per-token replay-input cache.

D LongBench Supplementary Analysis

Table 6 lists the complete macro-score grid underlying the main-text quality comparison. Recovery TTFT is reported in Section 5.2; Figure 8 pairs quality and timing on the same 28 questions. This supplement provides all-budget macro scores, paired CatchUp–Stale intervals, and per-task results at the common 10% budget. Fixed development checks compare Exact with a reference using the same prefill partition, verify complete-g_0 recovery at $W = D$, and check original token positions and isolation of unreplayed states.

Table 6: Complete LongBench quality grid on the same 139 questions (120 contexts). Scores are four-task macro means, 0–100. CatchUp budgets divide D; Tail-Replay budgets divide q and reuse the same observations across gap rows. Stale loads checkpoint states at p without replay.

				CatchUp (W/D)				Tail (K/q)	
Model	D	Exact	Stale	5%	10%	50%	100%	5%	10%
Qwen3.5-4B	1K	39.70	40.02	39.21	38.68	39.28	39.34	38.67	38.52
Qwen3.5-4B	2K	40.36	39.24	39.31	39.26	40.12	39.67	38.67	38.52
Qwen3.5-4B	3K	39.63	39.29	39.35	39.59	39.50	39.25	38.67	38.52
Qwen3.5-4B	4K	39.59	39.00	39.54	39.12	39.57	39.50	38.67	38.52
Qwen3.5-4B	8K	40.28	38.57	38.76	39.39	39.45	39.32	38.67	38.52
OLMo-Hybrid-7B	1K	35.66	35.03	35.01	34.61	34.69	34.93	24.33	25.68
OLMo-Hybrid-7B	2K	35.30	35.23	35.28	35.15	35.08	35.22	24.33	25.68
OLMo-Hybrid-7B	3K	35.31	34.56	34.69	34.68	35.44	34.82	24.33	25.68
OLMo-Hybrid-7B	4K	35.11	33.46	33.49	33.68	33.78	34.55	24.33	25.68
OLMo-Hybrid-7B	8K	35.20	36.65	35.76	35.34	35.68	35.51	24.33	25.68
Qwen3.6-27B	1K	47.62	46.70	46.25	46.66	46.34	46.67	44.38	44.37
Qwen3.6-27B	2K	47.49	46.26	46.34	46.39	46.44	46.67	44.38	44.37
Qwen3.6-27B	3K	47.49	44.40	44.92	44.30	45.00	44.27	44.38	44.37
Qwen3.6-27B	4K	46.38	46.04	44.98	46.01	45.72	45.91	44.38	44.37
Qwen3.6-27B	8K	47.44	45.73	46.40	45.61	47.39	47.30	44.38	44.37

D.1 No-replay ablation and checkpoint-gap sweep

Stale loads recurrent states saved at p and continues at q with matched FA KV, without replay. It isolates the effect of group-0 updates. Table 6 compares Exact, Stale, CatchUp, and Tail-Replay across all five checkpoint gaps and all evaluated budgets.

Effect of group-0 updates. At $D = 8$K on Qwen3.6-27B, 50% replay improves on Stale by 1.66 points (paired interval $[0.05, 3.81]$), recovering most of its 1.71-point deficit from Exact. Across the complete sweep, replay can also reduce scores; for example, Qwen3.5-4B at 1K with 10% replay loses 1.33 points relative to Stale. The tables below report paired intervals, and Section D.3 reports the separate quality–latency analysis on the 28 timing questions.

D.2 Paired CatchUp–Stale intervals

Table 7 uses 20,000 paired context-bootstrap resamples within tasks and identical draws across methods and gaps. Intervals are pointwise, without multiplicity adjustment. Task differences can offset one another in the macro mean.

D.3 Paired replay-budget tradeoff

The server has two A100 80GB GPUs. Each question's configurations use one GPU, with model loading and historical cache production separated from measurement. The runtime is PyTorch 2.13.0+cu132 with vLLM revision c55e15a44. Formal repetitions exclude additional diagnostic state-copying, hashing, and component-timing probes; the state transfer and private-state initialization required for recovery remain inside TTFT. Tail-Replay retains the largest evaluated tail's group inputs in pinned memory and transfers the requested slice. Latency statistics use per-input medians; repeated

Table 7: CatchUp minus Stale: macro-score differences and paired 95% context-bootstrap intervals.

D	5%	10%	50%	100%
Qwen3.5-4B				
1K	$-0.81\,[-2.19, 0.08]$	$-1.33\,[-2.80, -0.26]$	$-0.74\,[-1.96, -0.01]$	$-0.67\,[-1.90, 0.06]$
2K	$+0.07\,[-1.70, 1.99]$	$+0.02\,[-0.58, 0.63]$	$+0.88\,[-0.22, 2.56]$	$+0.43\,[-0.68, 2.11]$
3K	$+0.07\,[-1.09, 1.41]$	$+0.30\,[-0.92, 1.73]$	$+0.21\,[-1.06, 1.69]$	$-0.04\,[-1.27, 1.37]$
4K	$+0.54\,[-0.02, 1.31]$	$+0.12\,[-0.12, 0.39]$	$+0.57\,[0.04, 1.25]$	$+0.50\,[-0.28, 1.47]$
8K	$+0.19\,[-0.31, 0.70]$	$+0.81\,[-0.32, 2.44]$	$+0.87\,[-0.80, 2.67]$	$+0.74\,[-0.92, 2.52]$
OLMo-Hybrid-7B				
1K	$-0.02\,[-1.35, 1.06]$	$-0.42\,[-1.74, 0.65]$	$-0.34\,[-1.66, 0.74]$	$-0.10\,[-1.47, 1.03]$
2K	$+0.05\,[-0.11, 0.20]$	$-0.08\,[-0.66, 0.50]$	$-0.15\,[-0.38, 0.06]$	$-0.00\,[-0.33, 0.40]$
3K	$+0.13\,[-0.05, 0.42]$	$+0.13\,[-0.15, 0.46]$	$+0.89\,[-0.03, 2.29]$	$+0.27\,[-0.09, 0.79]$
4K	$+0.02\,[-1.05, 1.04]$	$+0.22\,[-0.94, 1.33]$	$+0.31\,[-0.79, 1.40]$	$+1.08\,[-0.46, 2.93]$
8K	$-0.89\,[-2.96, 0.67]$	$-1.31\,[-3.36, 0.25]$	$-0.96\,[-3.32, 1.23]$	$-1.14\,[-3.54, 1.15]$
Qwen3.6-27B				
1K	$-0.45\,[-1.23, 0.02]$	$-0.04\,[-0.15, 0.07]$	$-0.35\,[-1.10, 0.06]$	$-0.03\,[-0.16, 0.12]$
2K	$+0.08\,[-0.25, 0.45]$	$+0.13\,[-0.19, 0.52]$	$+0.19\,[-0.10, 0.53]$	$+0.42\,[0.04, 0.90]$
3K	$+0.53\,[-0.26, 2.25]$	$-0.09\,[-0.22, 0.02]$	$+0.60\,[-0.17, 2.33]$	$-0.12\,[-0.24, 0.01]$
4K	$-1.06\,[-2.62, -0.09]$	$-0.03\,[-0.15, 0.09]$	$-0.32\,[-0.88, 0.06]$	$-0.14\,[-0.59, 0.14]$
8K	$+0.66\,[-0.11, 1.72]$	$-0.12\,[-0.40, 0.08]$	$+1.66\,[0.05, 3.81]$	$+1.57\,[-0.05, 3.71]$

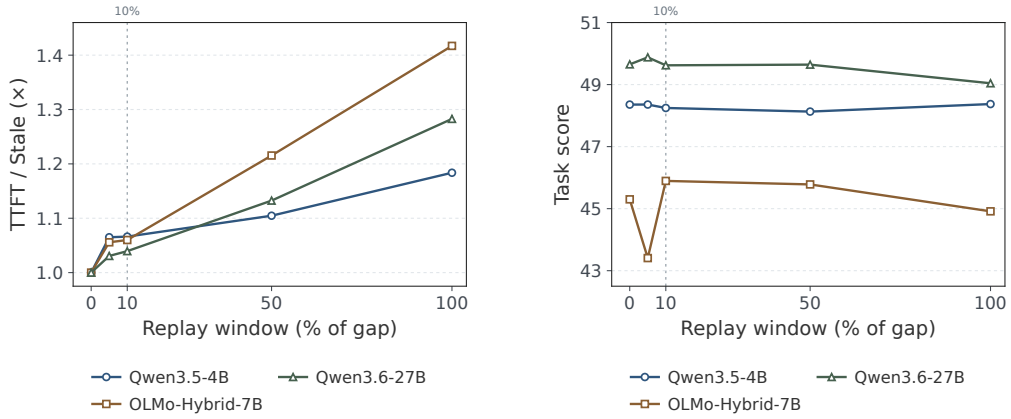

Figure 8: Replay budgets at $D = 8$K on the same 28 timed questions. Left: macro TTFT normalized by Stale. Right: macro task scores. Both panels place the evaluated budgets at their true numerical positions; the vertical guide marks 10%. Zero replay denotes Stale; all positive budgets update g_0 and retain deeper checkpoint states.

timings do not add independent quality samples. Figure 8 shows the complete budget sweep on the paired subset.

At the common 10% budget, paired TTFT ratios imply 3.1–7.5% added latency relative to Stale across the measured model–gap settings. At 8K, added mean TTFT is 7.9/7.5/12.1 ms and paired-cohort score changes are $-0.11, +0.59, -0.03$ for Qwen3.5-4B/OLMo-Hybrid-7B/Qwen3.6-27B. For Qwen3.6-27B at 8K with 50% replay, added paired TTFT is 13.2% and the same-cohort score change is -0.0085 points. The full 139-question gain of 1.66 points belongs to the separate quality cohort.

D.4 Answer-continuation sensitivity

On OLMo-Hybrid-7B, Tail-Replay's lower LongBench score is concentrated in HotpotQA and MuSiQue. Among their 76 outputs, 37 contain `Question:` under Tail-Replay 10%, compared with eight under Exact. A post-hoc sensitivity check extracts the first nonempty line for every method on these two tasks and leaves the other tasks unchanged. Table 8 shows the resulting four-task macro scores on all 139 questions at $D = 8$K. The Exact–Tail-Replay gap decreases from 9.52 to 1.66 points, and the CatchUp–Tail-Replay gap from 9.66 to 4.25 points, showing sensitivity to answer continuation under whole-output F1. The main results retain the original full-output scoring. In three full-prefix development checks with matched chunking, all 48 recurrent and convolution-state tensors matched the reference bitwise in each case.

Table 8: OLMo-Hybrid-7B output-scoring sensitivity. Replay fractions are 10% of D for CatchUp and q for Tail-Replay.

Scoring	Exact	CatchUp	Tail-Replay
Full output (main results)	35.20	35.34	25.68
First line on two QA tasks	37.47	40.06	35.81

D.5 Task-level results

Frozen selection and scoring. The quality cohort is the inherited, prespecified common-8K extension input list: 139 questions from 120 distinct contexts, with no resampling or score-based selection. It contains 36 NarrativeQA, 36 HotpotQA, 40 MuSiQue, and 27 QMSum questions. Their distinct-context counts are 18, 36, 40, and 26, respectively. The same questions are evaluated at every gap and for every model and method. The timing cohort is the original-order intersection of the earlier 50-question timing list with this quality cohort: 28 questions (7, 6, 10, and 5 by task) from 26 contexts. These frozen cohorts describe selected inputs, rather than the full LongBench test sets.

Tables 9–11 report per-task means on all 139 questions. NarrativeQA, HotpotQA, and MuSiQue use token F1; QMSum uses ROUGE-L. All entries multiply the native score by 100 and higher is better. The overall score is the equal-weight mean of the four task means, not the question-weighted mean. CatchUp uses $\lceil 0.1D \rceil$ replay tokens and Tail-Replay uses $\lceil 0.1q \rceil$. Each Tail-Replay observation is independent of D and is reused in every displayed gap row; repeated cells do not represent additional generations.

E BFCL Target-Turn Supplement

Frozen cohort construction. Histories are generated by Qwen3.5-4B, with one preset Base or Long-Context version per original scenario; both evaluated models receive the same text and build their own caches. Selection is independent of method scores and first-turn correctness: retain all 20 scenarios supporting 4K, then sample 20 of the remaining 22 supporting 2K with seed 20260922. The 40 scenarios comprise 35 Long-Context and five Base inputs. Eligibility requires $q = 1024\lfloor (r - 128)/1024 \rfloor$, $p = q - D > 0$, $r + 2048 \leq 32768$, and q inside the frozen history. Inputs are neither padded nor truncated. Eligible counts are 40/40/32/20 at 1/2/3/4K; cross-gap comparisons use their common 20-scenario intersection.

Table 9: Qwen3.5-4B per-task quality (0–100), using all 139 questions. Each task has the same cohort at every gap; Tail 10% denotes Tail-Replay.

Task / metric	D	Exact	Stale	CatchUp 10%	Tail 10%
NarrativeQA (F1)	1K	28.23	27.12	25.91	28.30
	2K	28.31	28.62	28.25	28.30
	3K	28.31	27.02	27.04	28.30
	4K	28.23	26.63	26.77	28.30
	8K	28.23	28.08	29.32	28.30
HotpotQA (F1)	1K	73.33	74.39	71.81	71.29
	2K	73.33	71.06	70.56	71.29
	3K	73.89	73.89	73.10	71.29
	4K	73.33	73.00	72.90	71.29
	8K	73.33	73.10	73.00	71.29
MuSiQue (F1)	1K	37.31	37.48	37.17	35.05
	2K	39.81	37.65	37.17	35.05
	3K	37.31	35.58	37.65	35.05
	4K	37.17	36.64	37.17	35.05
	8K	39.81	34.14	36.64	35.05
QMSum (ROUGE-L)	1K	19.94	21.09	19.85	19.43
	2K	19.98	19.62	21.06	19.43
	3K	19.03	20.66	20.57	19.43
	4K	19.61	19.74	19.64	19.43
	8K	19.74	18.97	18.58	19.43

Table 10: OLMo-Hybrid-7B per-task quality (0–100), using all 139 questions. Each task has the same cohort at every gap; Tail 10% denotes Tail-Replay.

Task / metric	D	Exact	Stale	CatchUp 10%	Tail 10%
NarrativeQA (F1)	1K	32.46	31.47	30.94	32.02
	2K	31.96	32.06	32.86	32.02
	3K	31.70	33.18	33.41	32.02
	4K	31.47	31.11	31.44	32.02
	8K	31.74	35.74	35.61	32.02
HotpotQA (F1)	1K	51.71	50.79	50.79	29.38
	2K	50.79	50.96	50.76	29.38
	3K	50.76	48.26	48.26	29.38
	4K	50.79	48.29	48.29	29.38
	8K	50.79	54.07	49.58	29.38
MuSiQue (F1)	1K	35.49	33.47	32.72	18.34
	2K	35.49	34.24	33.51	18.34
	3K	35.49	33.51	33.51	18.34
	4K	35.49	31.49	32.76	18.34
	8K	35.49	34.50	33.00	18.34
QMSum (ROUGE-L)	1K	22.97	24.39	24.00	22.98
	2K	22.99	23.65	23.46	22.98
	3K	23.29	23.27	23.55	22.98
	4K	22.71	22.97	22.23	22.98
	8K	22.78	22.28	23.18	22.98

Table 11: Qwen3.6-27B per-task quality (0–100), using all 139 questions. Each task has the same cohort at every gap; Tail 10% denotes Tail-Replay.

Task / metric	D	Exact	Stale	CatchUp 10%	Tail 10%
NarrativeQA (F1)	1K	36.33	36.33	36.33	27.63
	2K	36.33	35.87	36.33	27.63
	3K	36.33	30.59	30.59	27.63
	4K	34.67	33.19	33.19	27.63
	8K	36.33	34.02	34.02	27.63
HotpotQA (F1)	1K	73.38	73.38	73.38	72.75
	2K	72.75	72.47	72.75	72.75
	3K	73.38	72.75	72.75	72.75
	4K	72.75	73.38	73.38	72.75
	8K	73.38	73.38	73.38	72.75
MuSiQue (F1)	1K	59.38	55.63	55.63	56.88
	2K	59.38	55.63	55.63	56.88
	3K	59.38	53.13	53.13	56.88
	4K	56.88	56.88	56.88	56.88
	8K	59.38	54.38	54.38	56.88
QMSum (ROUGE-L)	1K	21.39	21.44	21.28	20.23
	2K	21.51	21.05	20.85	20.23
	3K	20.88	21.11	20.74	20.23
	4K	21.24	20.72	20.60	20.23
	8K	20.68	21.16	20.67	20.23

Checks and repeated measurements. Section 5.5 specifies generation limits and environment resets. Recovered state persists across tool calls. Success is the conjunction of official `state_checker` and `response_checker`; the latter matches target-turn reference tool-execution outputs against cumulative execution outputs through that turn. All 872 unique quality configurations passed CPU replay and re-scoring. Exact resumes its prescribed checkpoint at each gap; Tail-Replay shares observations across gaps with unchanged input and q.

Timing runs serially on the common 20 scenarios, using the same GPU for all configurations of a model/scenario: one compile request, three warmups, and five measurements per configuration (520 configurations, 2,600 formal measurements). Tokenization, cache production, tool execution, and network time are outside TTFT; required recovery transfers and initialization remain inside. For input medians $t_{i,m,D}$, absolute TTFT is $\frac{1}{20}\sum_i t_{i,m,D}$ and paired speedup is $\exp(\frac{1}{20}\sum_i \log[t_{i,\mathrm{Exact},D}/t_{i,m,D}])$. Intervals use 10,000 paired scenario resamples (seed 20260922), common draws, and pointwise percentiles. Repetitions do not add quality samples; zero-width intervals reflect coincident observed outcomes.

Tables 12–15 provide complete eligible-cohort success counts, paired intervals for both cohorts, and all-gap timing. Main Figure 6 compares Exact, CatchUp, and Tail-Replay on the common cohort; the tables here also include the no-replay Stale ablation.

Table 12: Complete eligible BFCL cohorts. Entries are successful target turns / scenarios; Both replay methods use 10% with their respective denominators.

Model	D	n	Exact	Stale	CatchUp	Tail-Replay
Qwen3.5-4B	1K	40	26/40	25/40	28/40	25/40
Qwen3.5-4B	2K	40	27/40	27/40	25/40	25/40
Qwen3.5-4B	3K	32	19/32	18/32	18/32	18/32
Qwen3.5-4B	4K	20	12/20	11/20	12/20	12/20
Qwen3.6-27B	1K	40	29/40	30/40	30/40	30/40
Qwen3.6-27B	2K	40	29/40	31/40	30/40	30/40
Qwen3.6-27B	3K	32	20/32	20/32	21/32	22/32
Qwen3.6-27B	4K	20	15/20	14/20	14/20	14/20

Table 13: CatchUp 10% minus each baseline on the common 20-scenario cohort. Entries are success-rate differences in percentage points [paired 95% interval]. Tail-Replay uses 10%.

Model	D	vs. Exact	vs. Stale	vs. Tail-Replay
Qwen3.5-4B	1K	$0\,[0, 0]$	$+5\,[0, +15]$	$0\,[-15, +15]$
Qwen3.5-4B	2K	$0\,[0, 0]$	$-5\,[-15, 0]$	$0\,[-15, +15]$
Qwen3.5-4B	3K	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[-15, +15]$
Qwen3.5-4B	4K	$0\,[-15, +15]$	$+5\,[0, +15]$	$0\,[0, 0]$
Qwen3.6-27B	1K	$0\,[0, 0]$	$0\,[0, 0]$	$+5\,[0, +15]$
Qwen3.6-27B	2K	$+5\,[0, +15]$	$0\,[0, 0]$	$+5\,[0, +15]$
Qwen3.6-27B	3K	$0\,[0, 0]$	$+5\,[0, +15]$	$0\,[0, 0]$
Qwen3.6-27B	4K	$-5\,[-15, 0]$	$0\,[0, 0]$	$0\,[0, 0]$

Table 14: CatchUp 10% minus each baseline on the complete eligible cohort at each gap. Entries are success-rate differences in percentage points [paired 95% interval]. Tail-Replay uses 10%.

Model	D	vs. Exact	vs. Stale	vs. Tail-Replay
Qwen3.5-4B	1K	$+5\,[-5, +15]$	$+7.5\,[0, +17.5]$	$+7.5\,[-5, +20]$
Qwen3.5-4B	2K	$-5\,[-12.5, 0]$	$-5\,[-12.5, 0]$	$0\,[-7.5, +7.5]$
Qwen3.5-4B	3K	$-3.12\,[-9.38, 0]$	$0\,[0, 0]$	$0\,[-9.38, +9.38]$
Qwen3.5-4B	4K	$0\,[-15, +15]$	$+5\,[0, +15]$	$0\,[0, 0]$
Qwen3.6-27B	1K	$+2.5\,[-5, +12.5]$	$0\,[0, 0]$	$0\,[-10, +10]$
Qwen3.6-27B	2K	$+2.5\,[-5, +10]$	$-2.5\,[-7.5, 0]$	$0\,[-7.5, +7.5]$
Qwen3.6-27B	3K	$+3.12\,[0, +9.38]$	$+3.12\,[0, +9.38]$	$-3.12\,[-9.38, 0]$
Qwen3.6-27B	4K	$-5\,[-15, 0]$	$0\,[0, 0]$	$0\,[0, 0]$

Table 15: BFCL timing on the common 20 scenarios. Method columns are mean per-scenario median TTFTs (ms). The final column is the geometric mean of paired Exact-to-CatchUp TTFT ratios [95% interval], computed from the per-scenario medians. Both replay methods use 10%.

Model	D	Exact	Stale	CatchUp	Tail-Replay	Speedup ($\times$)
Qwen3.5-4B	1K	171.4	109.3	117.3	157.4	$1.462\,[1.441, 1.480]$
Qwen3.5-4B	2K	217.6	108.8	117.3	157.4	$1.856\,[1.827, 1.884]$
Qwen3.5-4B	3K	265.5	110.0	117.2	157.4	$2.267\,[2.226, 2.308]$
Qwen3.5-4B	4K	306.8	109.3	117.3	157.4	$2.618\,[2.575, 2.663]$
Qwen3.6-27B	1K	546.4	261.0	270.6	402.2	$2.045\,[1.961, 2.124]$
Qwen3.6-27B	2K	794.9	261.5	270.3	402.2	$2.985\,[2.829, 3.134]$
Qwen3.6-27B	3K	1057.4	262.2	270.1	402.2	$3.979\,[3.736, 4.210]$
Qwen3.6-27B	4K	1286.5	261.6	271.3	402.2	$4.820\,[4.515, 5.112]$

F LONG-INPUT RECOVERY QUALITY

We use five RULER tasks: `niah_single_2`, `niah_multikey_1`, `vt`, `fwe`, and `qa_2`. Each model and input budget has 100 inputs per task. The two Qwen models share generated texts, whereas OLMo uses separately generated texts. Budgets include generation reserve: actual input lengths range from 13,264–16,330, 27,541–32,718, and 57,619–65,474 tokens in the 16K, 32K, and 64K conditions. Length cohorts are different inputs; within each model and length, all gaps use the same 500 inputs.

Reference implementation and scoring. The HF/official FLA implementation sets $q = 1024\lfloor (r - 128)/1024 \rfloor$ and $p = q - D > 0$. CatchUp uses $W = \lceil 0.1D \rceil$: 410/820/1,639/3,277 tokens for $D = 4/8/16/32$K. Reference directly installs the producer's state at q; it is a quality reference, unlike Exact, the native-vLLM checkpoint-resume path measured for TTFT in Section 5.1. Tail-Replay replays $\lceil 0.1q \rceil$ tokens using saved exact group inputs. Reference and Tail each run once per input; Stale and CatchUp run at each eligible gap. There are 36,000 method records over 4,500 model-input instances. Recorded full-generation times are not formal TTFT and are excluded from latency comparisons.

NIAH, variable tracing, and frequent-word extraction use the official multi-reference hit fraction; QA uses an any-reference match. Equal task sizes make the five-task macro mean equal to the pooled mean. We retain the original per-record rounded scores. Pointwise 95% percentile intervals directly bootstrap CatchUp–Stale differences, with 20,000 paired input resamples within each task and equal task weights. Intervals are pointwise and unadjusted for multiple comparisons; $[0, 0]$ records identical paired scores in the observed sample.

Table 16: Complete RULER five-task macro scores (0–100). Reference: producer state at q. CatchUp and Tail use 10% of D and q, respectively. Δ is CatchUp minus Stale; each row pairs the same 500 inputs. Reference and Tail observations are shared across gaps within a length.

Model	Budget	D	Reference	Stale	CatchUp	Tail	Δ [95% CI]
Qwen3.5-4B	16K	4K	92.67	92.73	92.73	92.93	$+0.00\,[0.00, 0.00]$
Qwen3.5-4B	16K	8K	92.67	92.69	92.89	92.93	$+0.20\,[0.00, 0.60]$
Qwen3.5-4B	32K	4K	92.20	92.60	92.36	93.40	$-0.24\,[-0.68, 0.00]$
Qwen3.5-4B	32K	8K	92.20	92.40	92.20	93.40	$-0.20\,[-0.60, 0.00]$
Qwen3.5-4B	32K	16K	92.20	92.56	92.76	93.40	$+0.20\,[0.00, 0.60]$
Qwen3.5-4B	64K	4K	91.40	91.20	91.20	91.33	$+0.00\,[0.00, 0.00]$
Qwen3.5-4B	64K	8K	91.40	91.00	91.20	91.33	$+0.20\,[0.00, 0.60]$
Qwen3.5-4B	64K	16K	91.40	91.00	91.20	91.33	$+0.20\,[0.00, 0.60]$
Qwen3.5-4B	64K	32K	91.40	91.40	91.60	91.33	$+0.20\,[-0.40, 0.80]$
OLMo-Hybrid-7B	16K	4K	87.80	88.15	88.23	87.69	$+0.08\,[-0.52, 0.68]$
OLMo-Hybrid-7B	16K	8K	87.80	87.84	87.64	87.69	$-0.20\,[-0.92, 0.44]$
OLMo-Hybrid-7B	32K	4K	87.12	86.87	86.89	87.07	$+0.03\,[-0.12, 0.20]$
OLMo-Hybrid-7B	32K	8K	87.12	86.61	86.39	87.07	$-0.23\,[-0.96, 0.45]$
OLMo-Hybrid-7B	32K	16K	87.12	86.79	86.63	87.07	$-0.16\,[-0.88, 0.48]$
OLMo-Hybrid-7B	64K	4K	85.97	85.61	85.68	81.53	$+0.07\,[-0.53, 0.67]$
OLMo-Hybrid-7B	64K	8K	85.97	85.61	86.11	81.53	$+0.49\,[-0.11, 1.28]$
OLMo-Hybrid-7B	64K	16K	85.97	85.21	85.23	81.53	$+0.01\,[-0.59, 0.60]$
OLMo-Hybrid-7B	64K	32K	85.97	83.25	83.36	81.53	$+0.11\,[-0.20, 0.43]$
Qwen3.6-27B	16K	4K	94.53	94.53	94.53	94.87	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	16K	8K	94.53	94.53	94.53	94.87	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	32K	4K	94.20	94.16	94.16	94.40	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	32K	8K	94.20	94.16	94.16	94.40	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	32K	16K	94.20	94.16	94.16	94.40	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	64K	4K	94.80	94.60	94.60	94.60	$+0.00\,[0.00, 0.00]$
Qwen3.6-27B	64K	8K	94.80	94.53	94.60	94.60	$+0.07\,[0.00, 0.20]$
Qwen3.6-27B	64K	16K	94.80	94.60	94.53	94.60	$-0.07\,[-0.20, 0.00]$
Qwen3.6-27B	64K	32K	94.80	94.40	94.33	94.60	$-0.07\,[-0.20, 0.00]$

No-replay ablation and task-level variation. At the 64K budget and 32K gap on OLMo-Hybrid-7B, CatchUp differs from Stale by $+0.11$ points (83.36 versus 83.25).

For OLMo-Hybrid-7B at a 64K budget and an 8K gap, the five-task macro comparison has three improved inputs, two regressions, and 495 ties. Across all 27 settings, CatchUp–Stale macro differences range from -0.24 to $+0.49$, and all paired intervals include zero.

G Additional Cohorts and Diagnostics

G.1 Original Five-Task Cohort and Qasper

The original frozen cohort has 40 questions each from NarrativeQA, Qasper, HotpotQA, MuSiQue, and QMSum. It precedes and overlaps the main 139-question cohort; these are not independent replications. The archived data contain all 10,152 original method records; no formal TTFT was collected. Qasper contributes 1,576 records, with eligible counts 38/33/23/19 at 1/2/3/4K for each Qwen model and 38/31/23/19 for OLMo; two questions support no tested gap. Table 17 reports token F1 (0–100) on their common 19-question intersection. Tail observations are shared across gaps; Exact uses its recorded result at each gap. This supplement does not change the main four-task macro score or timing cohort.

Table 17: Qasper on the common 19-question cohort. C denotes CatchUp and T denotes Tail-Replay; percentages have denominators D and q, respectively.

Model	D	Exact	Stale	C5%	C10%	T5%	T10%
Qwen3.5-4B	1K	34.91	34.15	34.45	33.97	38.07	33.31
Qwen3.5-4B	2K	34.91	35.30	34.27	33.97	38.07	33.31
Qwen3.5-4B	3K	35.71	40.95	35.96	40.61	38.07	33.31
Qwen3.5-4B	4K	34.59	40.97	40.65	40.80	38.07	33.31
OLMo-Hybrid-7B	1K	36.22	36.74	36.74	36.74	26.26	27.09
OLMo-Hybrid-7B	2K	34.92	35.94	34.44	36.01	26.26	27.09
OLMo-Hybrid-7B	3K	34.92	37.17	37.17	35.94	26.26	27.09
OLMo-Hybrid-7B	4K	36.02	33.18	33.18	33.18	26.26	27.09
Qwen3.6-27B	1K	46.12	45.03	45.03	45.03	47.34	47.34
Qwen3.6-27B	2K	45.03	45.89	45.15	46.20	47.34	47.34
Qwen3.6-27B	3K	46.12	45.89	46.93	46.43	47.34	47.34
Qwen3.6-27B	4K	46.12	48.01	44.84	45.87	47.34	47.34

G.2 Official-kernel recovery diagnostics

Input availability. In the separate 24-input official input-exactness diagnostic with Qwen3.5-4B, g_0's external token-derived replay inputs are bitwise identical to the reference in 24/24 cases. The diagnostic uses eight inputs each from NarrativeQA, HotpotQA, and QMSum, with $q = 8192$, $p = 6144$, and $W = 256$. Deeper contextual inputs have nonzero relative errors. This corroborates the input-availability distinction. Proposition 1 describes state-error propagation under shared gates, keys, and values within a single layer.

Evidence-position scope. A separate fixed-$D = 4$K diagnostic with Qwen3.5-4B moves cache boundaries for 97 LongBench questions, tracking answer mentions rather than complete multi-hop evidence. Questions are drawn from the LongBench base and E splits, deduplicated by context and question, and selected by answer-alias matching rather than task scores. The 62-question subset comprises all cases where a before-checkpoint boundary is feasible. CatchUp10 minus Stale is -0.37 F1 points ($[-1.12, 0.00]$, 97 questions) with the answer inside the replay window, and $+1.25$ ($[-0.25, 4.00]$, 62 eligible questions) with the answer before the checkpoint. Full-g_0 replay in the gap condition gives $+0.93$ ($[-1.47, 3.83]$, 97 questions). Intervals use 10,000 paired question resamples; the estimates are conditional on these cache boundaries and mention positions.

G.3 Runtime Environments

Native-vLLM experiments use PyTorch 2.13.0+cu132 and vLLM revision `c55e15a44`. RULER runs use a separate Hugging Face/FLA environment with PyTorch 2.9.1+cu128.