HISERVE: LOW-RANK PREFIX CACHING FOR HYBRID LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid language models combine recurrent layers with full attention to achieve competitive model quality while reducing the growth of the key/value (KV) cache. However, reusing shared prefixes requires separate recurrent-state checkpoints at cached boundaries. Storing these checkpoints in full increases memory pressure, eviction, and repeated prefill. Across five recurrent and hybrid language models, we find that recurrent states exhibit approximately low-rank structure, enabling substantial compression through truncated SVD. HiServe performs this compression asynchronously on the GPU and reconstructs states on reuse. At rank 16, HiServe provides a $3.98\times$ reduction in same-precision storage for the evaluated Qwen3.5 recurrent-state matrices, or $3.73\times$ including unchanged convolution states. Separate evaluations, including long-context tasks and periodic recompression, show benchmark-score changes from -0.63 to $+2.00$ percentage points. On multi-turn ShareGPT reference-history replay with Qwen3.5-4B on an NVIDIA GB10, HiServe improves throughput by 11.5% and reduces mean time to first token by 47.9% over SGLang's uncompressed prefix cache at a matched 15.23 GB cache budget. Gains depend on cache allocation and budget. Complementary trace-driven simulations with Marconi show a mean token cache-hit-rate improvement of 20.0 percentage points on ShareGPT under equal cache-memory budgets. Low-rank recurrent-state compression can thus increase prefix reuse and improve serving performance.

1 INTRODUCTION

State-space models (SSMs) (Gu et al., 2022; Gu & Dao, 2023; Dao & Gu, 2024) have emerged as a competitive alternative to attention-based architectures for sequence modeling. Their recurrent formulation offers two properties that are particularly attractive at deployment: per-token inference cost is independent of sequence length, and the memory footprint of the recurrent state is constant in the number of tokens processed. Hybrid language models, which interleave SSM layers with attention layers in a single stack, combine these efficiency properties with the modeling strengths of attention, and recent hybrids such as Zamba2 (Glorioso et al., 2024), Nemotron-H (NVIDIA et al., 2025), and Qwen3.5 (Qwen Team, 2026a) have demonstrated quality competitive with leading dense transformers at substantially reduced inference cost.

Prefix caching skips repeated prefill for shared prompts (Gim et al., 2024; Zheng et al., 2024b), but hybrid models complicate reuse (Pan et al., 2025). Attention KV entries are token-granular; recurrent states summarize complete prefixes and cannot be sliced to an earlier boundary. Each saved boundary therefore needs a separate checkpoint. In the evaluated Qwen3.5-4B configuration, 50.3 MB of FP32 recurrent matrices plus 1.18 MB of convolution state occupy 51.5 MB per checkpoint. One checkpoint for each of 64 prefixes costs 3.30 GB before attention KV storage. Under a fixed cache budget, these large checkpoints limit how many prefix boundaries can remain available for reuse, increasing eviction and repeated prefill. Marconi (Pan et al., 2025) manages which checkpoints to retain; we ask whether compressing each checkpoint can preserve more reusable prefixes within the same memory budget.

Our central finding is that recurrent states are approximately low rank. Although the recurrent-state dimensions of the models we study range from 64 to 256, most of the state energy, defined as the sum of squared singular values, is concentrated in a much smaller subspace. Across five models,

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.

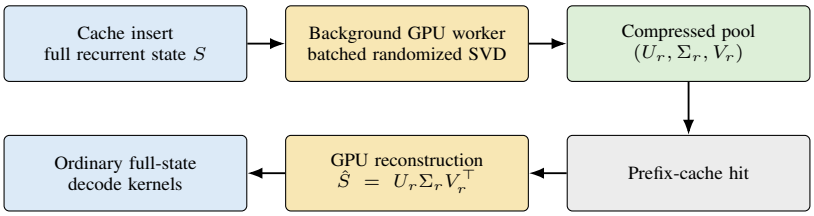

Figure 1: HiServe's dual-pool cache path. Compression runs asynchronously; a full-rank entry remains authoritative until its factors are committed. Cache hits reconstruct into the ordinary request working slot, leaving model kernels unchanged. The headline result uses multi-turn ShareGPT reference-history replay under a matched cache-memory budget.

the median effective rank required to retain 99% of the state energy ranges from only 2 to 24 (§4.2). This concentration suggests that storing every recurrent-state dimension at full rank is unnecessarily expensive for prefix caching.

We exploit this finding in **HiServe**, a prefix caching design that stores recurrent states as truncated low-rank factors. HiServe batches same-shaped states into a randomized SVD on the GPU and reconstructs full-rank states before reuse. Compression work is issued separately from request execution, and the full-rank entry remains authoritative until factorization completes. At rank 16, the factors provide a $3.98\times$ reduction in same-precision scalar storage for the evaluated Qwen3.5 recurrent-state matrices. Section 3 specifies the representation, accounting assumptions, and prototype scheduling.

At rank 16, median next-token KL is 1.4×10^{-5} on Nemotron-3-Nano-4B and 1.6×10^{-4} on Qwen3.5-4B for full-length prefixes. Separate rank-16 task evaluations show absolute accuracy differences of at most 2.0 percentage points. A trace-driven simulation of $4\times$ recurrent-state reduction raises mean ShareGPT hit rate by 20.0 percentage points under Marconi's policy. We also evaluate live serving with Qwen3.5-4B on an NVIDIA GB10 using reference-history replay of 64 ShareGPT sessions with 10 turns each. Compared with SGLang's uncompressed prefix cache at the same cache-memory budget, HiServe raises token cache hit rate from 3.28% to 67.15%, improves throughput by 11.5%, and reduces mean time to first token (TTFT) by 47.9%.

These results show that increasing recurrent-state cache capacity can materially improve serving under memory pressure. The live-serving result establishes gains at the evaluated cache budget and pool partition; it does not establish the same gains across allocations.

2 Background and Related Work

Recurrent and hybrid models. Recurrent models offer a route to efficient long-sequence inference by summarizing prior tokens in a fixed-size state rather than retaining a growing token-level KV cache. From structured SSMs such as S4 (Gu et al., 2022) to selective SSMs such as Mamba (Gu & Dao, 2023; Dao & Gu, 2024), this approach has enabled sequence modeling with bounded recurrent memory. Hybrid architectures combine recurrence with attention to balance inference efficiency and model quality (Lieber et al., 2024; Qwen Team, 2026a). These advances reduce memory growth within a sequence, but retaining many reusable prefixes still requires a separate recurrent-state snapshot at each cached boundary. HiServe targets this remaining storage cost through low-rank checkpoint compression.

Recurrent-state compression. Quamba and Quamba2 develop SSM-specific low-bit inference recipes (Chiang et al., 2025b;a). Q-Mamba explicitly quantizes state caches using decoupled scales and selectivity reconstruction (Chen et al., 2025); GHOST reduces Mamba2's state dimension through output-aware structured pruning (Menezes & Kyrillidis, 2026). Thus recurrent-state compression itself is not new. Our focus is per-prefix low-rank storage with asynchronous cache migration and restoration into unchanged full-state execution. The simple quantizers in our ablation do not reproduce these calibrated methods or establish superiority over them.

KV-cache and low-rank compression. KV methods exploit low precision (Liu et al., 2024), token selection (Zhang et al., 2023; Li et al., 2024), and attention sinks (Xiao et al., 2024). Token-selection methods need structure absent from a holistic recurrent state, whereas quantization remains applicable. Low-rank compression also predates this work in network weights (Denton et al., 2014). For attention caches, xKV jointly factorizes grouped layers (Chang et al., 2026), KQ-SVD targets attention fidelity (Lesens et al., 2025), and AttentionPack combines low-rank compaction with attention-aware decompression (Ilhan et al., 2026). HiServe applies factorization to saved recurrent states and measures the resulting retention/serving tradeoff, not a new SVD algorithm.

Prefix reuse and checkpoint placement. Prefix caching avoids repeated prefill by reusing computation for input tokens shared across requests, such as a common system prompt or conversation history. In attention-based models, SGLang's RadixAttention (Zheng et al., 2024b) finds the longest cached prefix and reuses its token-level KV entries, leaving only the unmatched suffix to process. State-space models complicate this reuse: a recurrent state summarizes an entire prefix and cannot be sliced back to an earlier token boundary. Reusing different prefix boundaries therefore requires separate state checkpoints, whose storage can limit cache capacity.

Prior work addresses which checkpoints to retain or how to recover missing states. Marconi (Pan et al., 2025) uses hybrid-aware admission and eviction to prioritize valuable checkpoints; sparse prefix caching (Shirokikh & Nikolenko, 2026) optimizes checkpoint placement and replays the remaining suffix. Tail-Replay (Liu et al., 2026) instead omits recurrent checkpoints and approximates states by replaying a short suffix. HiServe targets the size of the retained checkpoints directly: it compresses recurrent-state matrices into low-rank factors and reconstructs them on reuse, allowing more prefix boundaries to remain cached within the same memory budget.

3 HiServe Design

HiServe reduces prefix-cache storage by compressing recurrent-state checkpoints while leaving attention KV and convolution states unchanged. Built on SGLang's Mamba radix cache, it maintains separate pools for uncompressed and compressed checkpoints. Compression runs asynchronously, and the original checkpoint remains available until its compressed representation is ready. On reuse, HiServe reconstructs the recurrent state into a standard full-size working buffer before processing the remaining prompt or continuing generation. This allows cached states to use a compact representation while preserving the model's existing execution interface.

3.1 Truncated-SVD Prefix Encoding

For every recurrent layer and head, the state is a matrix $S \in \mathbb{R}^{D_k \times D_v}$, such as 128×128 in Qwen3.5-4B. HiServe encodes each matrix independently with a rank-r truncated SVD,

$$S \approx U_r \Sigma_r V_r^\top, \quad U_r \in \mathbb{R}^{D_k \times r}, \quad \Sigma_r \in \mathbb{R}^{r \times r}, \quad V_r^\top \in \mathbb{R}^{r \times D_v}.$$

We store a vector $s_r \in \mathbb{R}^r$ with $\Sigma_r = \mathrm{diag}(s_r)$, not a dense diagonal matrix. The factors require $r(D_k + D_v + 1)$ scalars instead of $D_k D_v$. At equal state/factor precision, the compression ratio is

$$\rho(D_k, D_v, r) = \frac{D_k D_v}{r(D_k + D_v + 1)}.$$

For the evaluated Qwen3.5 recurrent-state matrices ($D_k = D_v = 128$) at $r = 16$, $\rho = 3.98\times$. The ratios in Table 1 use this same-precision accounting and exclude the unchanged convolution state and attention KV cache.

3.2 System Architecture

SGLang's recurrent prefix cache stores checkpoints in fixed-size GPU state slots. HiServe extends this design with two pools: an uncompressed pool for working states and full-state checkpoints, and a compressed pool for low-rank checkpoints. Attention KV storage and convolution states remain unchanged.

An asynchronous GPU worker compresses retained checkpoints from the uncompressed pool, deferring new compression batches while prefill is pending or in progress. The original checkpoint remains

available until the scheduler installs the compressed representation and releases its slot. When a request reuses a compressed prefix, HiServe reconstructs its recurrent state directly into a full-size working slot before processing the remaining prompt or continuing generation. Inference therefore uses the existing model kernels.

The worker batches same-shaped recurrent states and applies randomized SVD (Halko et al., 2011) with four oversampling dimensions and one power iteration. Our implementation draws on ideas from the KV-SVD project (bairixie, 2026) for hardware-efficient randomized SVD. Bounded staging buffers limit outstanding snapshots and hold completed factors awaiting installation. These buffers count toward the cache-memory budget used in Section 5; Appendix D provides scheduling and allocation details. Full-size working slots and staging memory consume part of this budget, so checkpoint compression does not translate directly into an equal increase in cache capacity.

4 Evaluation

4.1 Setup and Methodology

We evaluate state reconstruction and next-token prediction fidelity across five recurrent and hybrid models, followed by downstream task accuracy. Deterministic full SVD provides reference spectra for effective rank and energy; reconstruction and prediction-fidelity experiments use randomized SVD (Halko et al., 2011). Appendix A gives checkpoint identifiers and protocols. Section 5 evaluates serving performance.

4.2 SSM State is Empirically Low-Rank

A single *state* denotes the recurrent-state matrix of one head within one recurrent layer. The five models in Table 1 span pure and hybrid architectures, including SSMs, DeltaNet (Yang et al., 2024), and Gated DeltaNet (Yang et al., 2025); Appendix A gives their state counts.

To sample a diverse input distribution, we draw 200 prompts from ShareGPT (ShareGPT contributors, n.d.) and 200 code prompts from CodeParrot (CodeParrot contributors, n.d.). For each prompt we record the recurrent state at fixed sequence positions (256, 512, 1,024, 2,048, and 4,096 tokens) and at the final token of the full prompt. We also generate a random control matrix of identical shape for each state, sampled from a Gaussian distribution with variance matched to the element-wise variance of the original.

Effective rank. Table 1 shows median effective ranks of 2–24 at 99% retained energy,[1] compared with 60–198 for the Gaussian controls. Because the controls match each state's shape and entry variance, this gap indicates structure beyond these basic matrix properties. DeltaNet has the largest median. Figure 2b shows the concentration at lower ranks, with a substantial upper tail for Gated DeltaNet.

[1]The effective rank at threshold τ is the smallest k such that $\sum_{i=1}^{k} \sigma_i^2 \geq \tau \sum_{i=1}^{n} \sigma_i^2$, where $n = \min(D_k, D_v)$ and $\sigma_1 \geq \cdots \geq \sigma_n$.

Table 1: SSM states use fewer effective dimensions than matched random matrices, but truncation error has model-dependent tails. Over 400 prompts at 6 lengths, we report effective rank at 99% energy (median and p_{99}), a random control, and rank-16 mean cosine similarity and squared relative Frobenius error.

		Eff. rank @99%		Control	Rank-16 reconstruction			
Model	State ($D_k \times D_v$)	med	p_{99}	med	cos (mean)	relFro^2 (med)	relFro^2 (p_{99})	Mem. red.
Mamba2-1.3B	64×128	5	33	60	0.9982	1.4×10^{-4}	5.8×10^{-2}	62.3%
Nemotron-3-Nano-4B	80×128	2	13	72	0.9997	5.8×10^{-5}	6.6×10^{-3}	67.3%
Qwen3.5-4B	128×128	9	48	99	0.9945	2.2×10^{-3}	1.1×10^{-1}	74.9%
DeltaNet-1.3B	128×128	24	77	99	0.9811	1.9×10^{-2}	2.3×10^{-1}	74.9%
GatedDeltaNet-1.3B	256×256	10	146	198	0.9619	6.5×10^{-4}	4.6×10^{-1}	87.5%

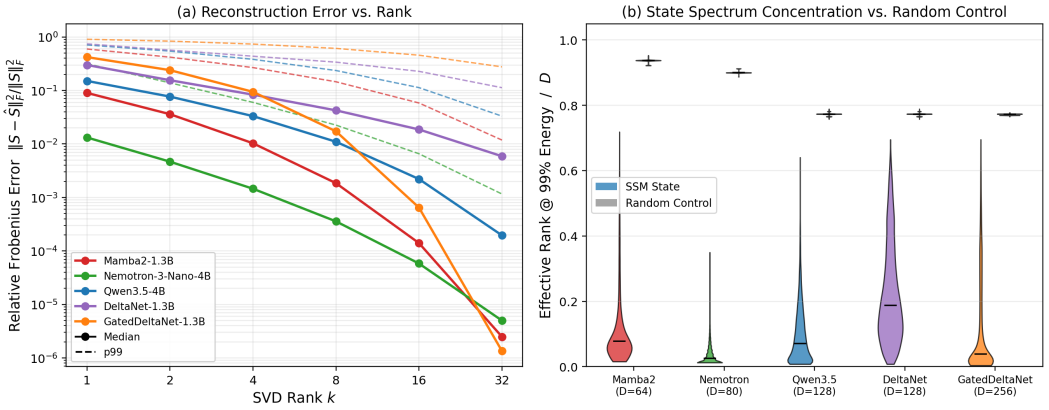

Figure 2: Low-rank structure of SSM states. **(a)** Squared relative SVD reconstruction error $\|S - \hat{S}_k\|_F^2/\|S\|_F^2$ vs. truncation rank k (solid: median, dashed: p_{99}). **(b)** Effective rank at 99% energy, normalized by state dimension D, for SSM states vs. a matched random-matrix control. States occupy far fewer effective dimensions than random matrices of the same size.

Reconstruction error. Figure 2a shows reconstruction error as a function of truncation rank. At rank 16, median squared relative Frobenius error ranges from 5.8×10^{-5} for Nemotron to 1.9×10^{-2} for DeltaNet. Tails are substantially larger: the 99th-percentile error is about 0.23 for DeltaNet and 0.46 for Gated DeltaNet. This spread shows that a fixed rank yields heterogeneous reconstruction quality across states and architectures. The downstream impact of the larger errors in DeltaNet and Gated DeltaNet remains untested.

4.3 Output Distribution Fidelity

We evaluate how low-rank compression changes next-token predictions and how its storage–fidelity trade-off compares with per-head affine INT8. Across five models, we measure $D_{\mathrm{KL}}(p_{\mathrm{full}} \| p_{\mathrm{probe}})$ in nats; lower values indicate predictions closer to the uncompressed reference.

Setup. For each of 100 ShareGPT and 100 CodeParrot prompts, we generate separate 128-token greedy reference continuations from a 256-token prefix and the full prompt, capped at 65,536 tokens. Every four decoding steps, each method compresses and reconstructs the same reference recurrent states, jointly modifying all recurrent heads and layers while leaving attention KV and convolution states unchanged. After measuring next-token KL, we discard the probe caches and continue the unmodified reference trajectory. This yields 6,400 paired probes per model, prefix condition, and method, measuring sensitivity to a single compression event without accumulating approximation error.

Storage reductions are relative to FP32 recurrent matrices and exclude unchanged KV and convolution states. SVD stores FP32 factors as described in §3.1; INT8 stores one byte per matrix element plus eight bytes of metadata per head, giving approximately $4\times$ compression. SVD's storage reduction depends on matrix shape and rank. Table 2 reports full-prompt results at rank 16; Figure 3 and Appendix A.1 provide the complete rank sweeps.

Fidelity and storage gains on the evaluated hybrids. On Qwen3.5-4B, rank-16 SVD provides $3.98\times$ compression and reduces median KL by 59–75% and p_{99} KL by 64–80% relative to INT8 across the two prefix conditions. At nearly equal storage, SVD therefore preserves predictions more closely for both the typical probe and the upper tail of the probe distribution.

Lower ranks can trade this fidelity advantage for smaller checkpoints. On Qwen, rank-8 SVD provides $7.97\times$ compression, using about half the recurrent-matrix storage of INT8. At 256 tokens, it reduces median KL from 1.01×10^{-3} to 8.17×10^{-4} and p_{99} KL from 0.177 to 0.129. On full prompts, however, median KL increases from 3.98×10^{-4} to 5.06×10^{-4}, while p_{99} KL falls from 0.129 to 0.103. Rank 16 improves both metrics in both prefix conditions.

Table 2: Paired full-prompt prediction fidelity at rank 16 versus per-head affine INT8. Each condition has 6,400 probes; KL is in nats. Lower KL is better; storage is not matched across all models.

	SVD rank 16		Per-head affine INT8	
Model	Median KL	p_{99} KL	Median KL	p_{99} KL
Qwen3.5-4B	1.63e−4	0.0460	3.98e−4	0.129
Nemotron-3-Nano-4B	1.39e−5	6.39e−3	0.0292	6.59
GatedDeltaNet-1.3B	0.0514	3.05	2.15e−4	9.78e−3
Mamba2-1.3B	3.46e−3	0.264	0.0377	1.70
DeltaNet-1.3B	0.0920	2.86	3.72e−3	0.402

Table 3: Native versus rank-16 task scores (%); compression follows prefill and every 1,024 output tokens. GPQA-Diamond: means (sample SD) over four paired seeds for each model. MMLU-Pro: scores (harness SE). LongBench v2: one run per condition.

	Qwen3.5-35B-A3B		Nemotron-3-Nano-30B-A3B	
Benchmark	Native	Compressed	Native	Compressed
LongBench v2	55.50	57.50	42.90	42.90
GPQA-Diamond	81.82 (1.09)	81.19 (0.64)	65.53 (2.81)	65.91 (3.02)
MMLU-Pro	84.67 (0.32)	85.01 (0.32)	76.27 (0.40)	76.48 (0.39)

Nemotron-3-Nano-4B supports a favorable trade-off in both prefix conditions: rank-8 SVD provides $6.12\times$ compression while reducing both median and p_{99} KL relative to INT8. Together, these results show that low-rank encoding can improve prediction fidelity at comparable storage or reduce storage at comparable or better fidelity, depending on the model, rank, and prefix condition.

Model dependence. These advantages do not extend uniformly to the other recurrent models. GatedDeltaNet favors INT8 even at nearly equal storage: rank-32 SVD provides $3.99\times$ compression but has full-prompt median KL of 0.0120, versus 2.15×10^{-4} for INT8. DeltaNet also favors INT8 on full prompts at approximately matched storage: rank-16 SVD gives median KL of 0.0920, versus 0.00372. Rank-16 median KL is about $8.4\times$ higher on full prompts than at 256 tokens for DeltaNet and $5.0\times$ higher for GatedDeltaNet.

Mamba2 gives mixed results despite rank-16 SVD using more storage than INT8: SVD has lower median KL on full prompts but higher median KL at 256 tokens. The consistent storage–fidelity advantage over the tested per-head affine INT8 quantizer is therefore confined to the two evaluated hybrid models.

4.4 Downstream Benchmarks

We evaluate task accuracy under periodic SVD-based recurrent-state compression on MMLU-Pro (Wang et al., 2024), LongBench v2 (Bai et al., 2025), and GPQA-Diamond (Rein et al., 2024), using Qwen3.5-35B-A3B (Qwen Team, 2026b) and Nemotron-3-Nano-30B-A3B (NVIDIA, 2025). We reconstruct all recurrent states after prefill and every additional 1,024 generated tokens, leaving attention KV and convolution states unchanged. This tests generation quality under repeated truncation. Appendix B gives protocol and statistical details.

Across the evaluated benchmarks, SVD compression largely preserves observed task accuracy, with score differences ranging from -0.63 to $+2.00$ percentage points (Table 3). LongBench v2 tests long-context understanding and reasoning (Bai et al., 2025), and neither model shows an accuracy drop after SVD compression in our evaluated runs. For GPQA-Diamond, paired-seed mean differences and approximate 95% confidence intervals are -0.63 $[-2.85, 1.58]$ points for Qwen and $+0.38$ $[-6.09, 6.85]$ for Nemotron ($n = 4$ seeds each). These intervals describe seed variability and do not establish noninferiority. MMLU-Pro's reported errors are per-condition harness standard errors, not paired-difference intervals; LongBench v2 provides one run per condition.

Appendix C compares SVD with per-head uniform quantization on Qwen3.5-4B using GSM8K (Cobbe et al., 2021). These experiments compress and restore the recurrent state once after prefill, then continue generation without further compression.

5 System Evaluation

We evaluate whether the additional checkpoint capacity enabled by HiServe reduces repeated prefill and improves serving performance under a matched cache-memory budget. The live comparison uses Qwen3.5-4B with rank-16 SVD on one NVIDIA GB10 (DGX Spark) and the architecture in Section 3.2. We first connect cache retention to measured performance, then examine cache reuse on broader workloads using trace simulation.

5.1 Matched-Memory Protocol

Each run replays 64 ShareGPT conversations with 10 turns each, for 640 requests. Later prompts include the complete recorded history, and each response is generated for the recorded assistant reply's token count with EOS stopping disabled. Both modes receive identical prompts and target output lengths; generated text does not replace the reference history. There is no benchmark context cap, response-length cap, or token truncation.

Both modes use SGLang's extra-buffer recurrent-state tracking with a 256-token interval. Working states, tracking buffers, and retained checkpoints share the full-state pool; checkpoints are saved during prefill and decoding as conversations progress. Working states are released when requests finish, while retained checkpoints remain until eviction. Thus 64 sessions can create many more than 64 checkpoints, which compete with the buffers needed to execute requests. We defer new compression while prefill work is pending or executing to avoid competing for GPU resources when prefill saturates the device.

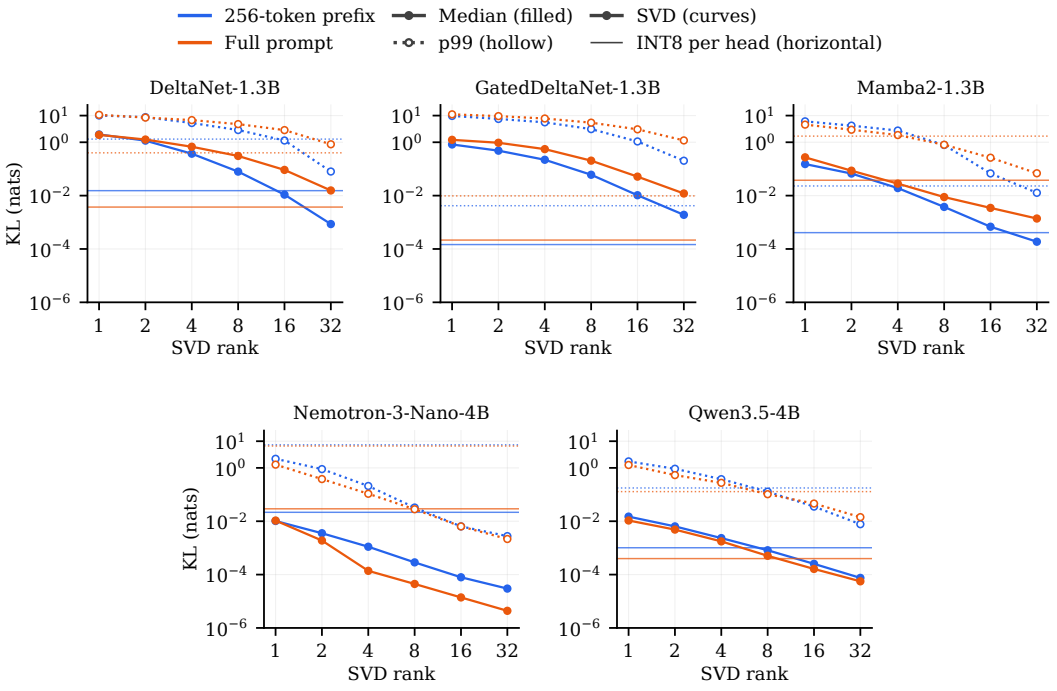

Figure 3: Median and p_{99} next-token KL (nats) across SVD ranks and prefix conditions. Horizontal lines give the per-head affine INT8 baseline on the same reference states; they are not storage-matched SVD comparisons. The hybrids retain favorable SVD comparisons across both prefix conditions; the pure linear-attention models show higher rank-16 SVD KL on full prompts. Percentile curves describe probe variability, not confidence intervals.

Table 4: Multi-turn ShareGPT reference-history replay at a matched 15.23 GB cache ceiling. Off/on values are means across five repetitions; changes are paired on-minus-off differences. Token hit rate is cached input tokens divided by all input tokens; recomputed prefix tokens measure missed reuse of prior-input prefixes.

Metric	Off	On	Change
Token hit rate (%)	3.28	67.15	63.87 pp
Recomputed prefix tokens	8.32e5	1.42e5	−6.89e5
Output throughput (tokens/s)	128.07	142.75	14.68
Mean TTFT (ms)	482.5	251.5	−231.0
P50 TTFT (ms)	396.2	206.1	−190.1
P95 TTFT (ms)	1073.2	506.8	−566.4
P99 TTFT (ms)	1686.3	871.3	−815.0
Mean TPOT (ms)	59.56	54.08	−5.47

We compare both modes under a 15.23 GB cache-memory budget, with KV capacity fixed at 262,144 tokens. Compression reduces each checkpoint from 51.5 MB to 13.8 MB, a $3.73\times$ reduction including unchanged convolution state. Within this budget, the baseline allocates 128 full-state slots, while HiServe allocates 32 full-state slots and 298 compressed slots and reserves 0.82 GB for snapshots and compressed results awaiting installation. The budget includes KV storage, state pools, and this staging memory. Appendix D provides the detailed accounting and tracking configuration.

5.2 Serving Results

Compression converts smaller checkpoints into useful prefix reuse. Under the matched cache budget, retaining more checkpoints avoids 82.9% of the prefix tokens recomputed by the uncompressed baseline (Table 4). The storage reduction therefore preserves states that later requests actually reuse, reducing repeated prefill rather than merely increasing nominal cache capacity.

The larger reduction in mean TTFT (47.9%) than the gain in output throughput (11.5%) is consistent with this mechanism. Checkpoint reuse primarily removes repeated prefill before the first output token, whereas throughput includes the time spent generating the full response. Both modes generate the same target output lengths, so this remaining decoding work limits the end-to-end gain from avoiding prefill. Lower P95 and P99 TTFT, with paired difference intervals below zero, provide empirical evidence that asynchronous SVD of recurrent states does not introduce a net tail-latency penalty under the tested load: the reuse benefits outweigh its overhead even in the measured latency tails.

Asynchronous compression also produces checkpoints in time for useful reuse under the tested load. By the end of measurement, 99.91% of enqueued compression jobs have completed, and requests restore compressed states about 470 times per run (Appendix E). The completion fraction shows that the worker keeps up with the submitted compression work; the restores establish that its outputs become usable checkpoints during serving. Together, these observations support asynchronous compression as a practical way to realize the additional cache capacity in this workload.

Memory cost. Peak accounted cache-state memory is 14.85 GB, below the 15.23 GB ceiling. Peak process CUDA allocation is about 25.21 GB in both modes; the paired difference is -0.40 MB (95% CI $[-85.95, 85.14]$ MB). This interval does not establish equivalence, but the point estimate shows no net increase in peak process allocation. The comparison matches accounted cache memory, not total process memory; Appendix E reports retention and worker diagnostics.

5.3 Trace-Driven Cache Simulation

To test cache reuse across additional workloads and cache budgets, we replay LMSYS-Chat-1M (Zheng et al., 2024a), ShareGPT (ShareGPT contributors, n.d.), and SWE-bench (Jimenez et al., 2024) traces in the Marconi simulator (Pan et al., 2025). Each comparison holds the trace, byte budget, arrival rate, and policy family fixed, and divides recurrent-matrix storage cost by four relative to its uncompressed baseline; KV and convolution costs are unchanged.

Table 5: Mean simulated token hit-rate improvement from $4\times$ recurrent-matrix compression at fixed cache budgets, in percentage points. Differences are computed at matching grid settings within each policy and averaged uniformly over 30 LMSYS, 36 ShareGPT, and 72 SWE-bench configurations. KV and convolution-buffer storage remain unchanged.

Workload	Admit-all	Selective	Marconi	Best-static
LMSYS-Chat-1M	+0.7	+9.8	+11.9	+6.9
ShareGPT	+10.7	+21.0	+20.0	+17.3
SWE-bench	+1.3	+7.0	+4.0	+3.9

We cross six session arrival rates (0.25, 0.5, 1, 2, 5, and 10 sessions/s) with workload-specific cache-memory budgets: 1, 2, 3, 4, and 5 GB for LMSYS-Chat-1M; 1, 2, 3, 4, 5, and 10 GB for ShareGPT; and 40, 60, 80, and 100 GB for SWE-bench. SWE-bench additionally varies the average time between requests within a session over 5, 7.5, and 10 seconds. These grids yield 30 LMSYS, 36 ShareGPT, and 72 SWE-bench configurations per policy, spanning different cache capacities and offered loads.

We evaluate four cache insertion policies, Admit-all, Selective, Marconi, and Best-static, comparing each compressed configuration with its own uncompressed policy baseline. Table 5 averages compressed-minus-uncompressed token hit-rate differences uniformly across each workload's grid. Appendix E.2 defines the policies and parameter-selection details.

Mean token hit-rate lift is positive for every policy/workload combination, including 20.0 percentage points for ShareGPT under Marconi (Table 5). The simulation supports increased cache reuse under reduced storage costs, extending the evidence for compact recurrent-state caching to conversational and software-engineering workloads.

6 Conclusion

Recurrent states across five models exhibit approximately low-rank structure that supports substantial compression through truncated SVD. HiServe applies this finding to prefix caching, storing compact factors and reconstructing states on reuse. Separate quality evaluations, including long-context tasks and periodic recompression, show small observed score differences. Serving experiments demonstrate a practical benefit: at 15.23 GB on GB10, throughput improves 11.5% and mean TTFT falls 47.9% on multi-turn ShareGPT reference-history replay.

The quality evidence is limited to the evaluated models, ranks, and tasks. The live-serving evidence covers one cache budget and pool partition. Future work includes dynamic pool resizing to reduce stranded capacity and serving evaluation with model-generated conversation histories; these extensions remain untested.

AI USE STATEMENT

We used generative AI tools to assist with implementation and limited aspects of experiment design. Experiment design was primarily guided by the authors, with AI providing supporting assistance. We also used generative AI tools to assist with writing and editing the manuscript and adapting its LaTeX formatting to the ICLR template. We reviewed and checked all AI-assisted outputs. We take responsibility for the final content of this work, including the text, code, experimental design, and claims produced with the aid of generative AI.

REFERENCES

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proc. ACL*, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

bairixie. KV-SVD: Hardware-efficient randomized SVD for online LLM KV-cache compression. GitHub repository, 2026. URL https://github.com/bairixie/kv-svd/tree/f4780d91cfe93c15e087ae82f2cc8ca1f4f13aee. Reference snapshot f4780d9, May 3, 2026. Accessed September 25, 2026.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-layer KV-cache compression via aligned singular vector extraction. In *Proc. ICML*, 2026. URL https://arxiv.org/abs/2503.18893v2.

Tianqi Chen, Yuanteng Chen, Peisong Wang, Weixiang Xu, Zeyu Zhu, and Jian Cheng. Q-Mamba: Towards more efficient Mamba models via post-training quantization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10594–10610, 2025. doi: 10.18653/v1/2025.findings-acl.551. URL https://aclanthology.org/2025.findings-acl.551/.

Hung-Yueh Chiang, Chi-Chih Chang, Natalia Frumkin, Kai-Chiang Wu, Mohamed S. Abdelfattah, and Diana Marculescu. Quamba2: A robust and scalable post-training quantization framework for selective state space models. *arXiv preprint arXiv:2503.22879*, 2025a. URL https://arxiv.org/abs/2503.22879.

Hung-Yueh Chiang, Chi-Chih Chang, Natalia Frumkin, Kai-Chiang Wu, and Diana Marculescu. Quamba: A post-training quantization recipe for selective state space models. In *Proc. ICLR*, 2025b. URL https://openreview.net/forum?id=mnna9LUg7P.

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

CodeParrot contributors. GitHub Code Dataset. https://huggingface.co/datasets/codeparrot/github-code, n.d. Accessed September 16, 2026.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proc. ICML*, 2024.

Remi Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Proc. NeurIPS*, 2014.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. Zenodo, version v0.4.0, 2023. URL https://zenodo.org/records/10256836. doi: https://doi.org/10.5281/zenodo.10256836.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://arxiv.org/abs/2311.04934.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, Anna Golubeva, Vasudev Shyam, James Whittington, Jonathan Pilault, and Beren Millidge. The Zamba2 suite: Technical report. *arXiv preprint arXiv:2411.15242*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *Proc. ICLR*, 2022.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011.

Fatih Ilhan, Gaowen Liu, Ramana Rao Kompella, Selim Furkan Tekin, Tiansheng Huang, Zachary Yahn, Yichang Xu, and Ling Liu. Attention-aware inference optimizations for large vision-language models with memory-efficient decoding. *arXiv preprint arXiv:2603.23914*, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *Proc. ICLR*, 2024.

Damien Lesens, Beheshteh T. Rakhshan, and Guillaume Rabusseau. KQ-SVD: Compressing the KV cache with provable guarantees on attention fidelity. *arXiv preprint arXiv:2512.05916*, 2025.

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. In *Proc. NeurIPS*, 2024.

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.

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:2608.30310*, 2026. URL https://arxiv.org/abs/2608.30310.

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. In *Proc. ICML*, 2024.

Michael Menezes and Anastasios Kyrillidis. GHOST: Unmasking phantom states in Mamba2 via grouped hidden-state output-aware selection & truncation. *arXiv preprint arXiv:2602.11408*, 2026. URL https://arxiv.org/abs/2602.11408.

NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025. URL https://arxiv.org/abs/2512.20848.

NVIDIA et al. Nemotron-H: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

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. *Proceedings of Machine Learning and Systems*, 7, 2025.

Qwen Team. Qwen3.5-4B. Hugging Face model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-4B.

Qwen Team. Qwen3.5-35B-A3B. Hugging Face model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.5-35B-A3B.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

ShareGPT contributors. ShareGPT. Project repository, n.d. URL https://github.com/domeccleston/sharegpt-monorepo. Accessed September 16, 2026; project attribution, not a versioned dataset snapshot.

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.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3018. Datasets and Benchmarks Track.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *Proc. ICLR*, 2024.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proc. NeurIPS*, 2024. URL https://arxiv.org/abs/2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *Proc. ICLR*, 2025. URL https://arxiv.org/abs/2412.06464.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Proc. NeurIPS*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset. In *Proc. ICLR*, 2024a.

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 *Proc. NeurIPS*, 2024b.

A State Characterization and KL Protocols

Model checkpoints. The five-model analysis uses `Qwen/Qwen3.5-4B`, `AntonV/mamba2-1.3b-hf`, `unsloth/NVIDIA-Nemotron-3-Nano-4B`, `fla-hub/delta_net-1.3B-100B`, and `m-a-p/1.3B-100B-GatedDeltaNet-pure`.

Architectures and state counts. Each forward pass maintains 768 recurrent states in Qwen3.5-4B (hybrid GDN-Transformer), 3,072 in Mamba2-1.3B (pure SSM), 2,016 in Nemotron-3-Nano-4B (hybrid Mamba2-Transformer), 384 in DeltaNet-1.3B (pure linear attention), and 192 in GatedDeltaNet-1.3B (pure gated linear attention). Each state corresponds to one head in one recurrent layer; Table 1 gives the matrix dimensions.

State-characterization precision. The GPU execution path uses BF16 model weights. Captured states are converted to float64 for full-SVD effective-rank and energy statistics. Reconstruction uses randomized SVD with four oversampling dimensions and one power iteration in float64. We disable Qwen3.5's fused causal-convolution path during state capture to expose the recurrent cache.

Spectrum and reconstruction. We use 200 ShareGPT (ShareGPT contributors, n.d.) chat prompts and 200 CodeParrot (CodeParrot contributors, n.d.) code prompts. Consecutive source documents are concatenated to at least 24,000 characters, then tokenized separately for each model. We sample states at 256, 512, 1,024, 2,048, and 4,096 tokens and at each prompt's full length, capped at 65,536 tokens. Each observation identifies a prompt, sequence position, layer, and head. The Gaussian control matches the state's shape and element-wise variance. Squared relative Frobenius error is $\|S - \hat{S}\|_F^2 / \|S\|_F^2$.

Next-token KL. The five-model comparison uses 100 ShareGPT and 100 CodeParrot prompts at 256-token and full-prompt boundaries, with a 65,536-token safety cap. Each condition has a separate 128-token greedy reference trajectory, probed every four steps. All methods compress identical reference states across all recurrent heads and layers; attention KV and convolution states remain unchanged. Probe caches are discarded. We compare randomized SVD at ranks 1, 2, 4, 8, 16, and 32 with affine min/max INT8 using one scale and zero point per head. SVD ranks 4/8/16 share a rank-16 sketch with eight oversampling dimensions; ranks 1/2/32 share a rank-32 sketch with zero oversampling. Both use two power iterations.

A.1 Complete Prediction-Fidelity Sweeps

Tables 6–10 report all six SVD ranks and the per-head affine INT8 baseline for both prefix conditions. Each row aggregates 6,400 probes; "Full" denotes the complete prompt (prefix -1 in the source results). KL is in nats, top-1 is agreement with the reference argmax, and reconstruction error is mean per-head squared relative Frobenius error averaged over probes. Percentiles use nearest rank. These are paired probes of identical reference states, not independent runs. We report descriptive aggregate statistics without prompt-level paired confidence intervals or significance tests.

For nominal recurrent-matrix storage, we count $4D_kD_v$ bytes for an FP32 original and $4r(D_k + D_v + 1)$ bytes for FP32 SVD factors, including the singular-value vector. INT8 assumes one packed byte per element and two four-byte metadata values per head (scale and zero point), giving $D_kD_v + 8$ bytes. This is explicit storage accounting, not measured cache allocation; it excludes attention KV, convolution state, padding, allocator overhead, and temporary workspaces. Without quantization metadata, INT8 is exactly $4\times$; including the assumed eight bytes makes it slightly smaller. Fixed SVD rank does not match storage across matrix shapes.

Table 6: Qwen3.5-4B: paired single-step prediction fidelity for 128×128 recurrent matrices. Storage is nominal reduction relative to FP32; INT8 uses the per-head affine baseline.

Prefix	Method	Storage	Median KL	p_{99} KL	Top-1 (%)	relFro^2
256	SVD 1	63.75×	0.0149	1.75	91.64	0.191
	SVD 2	31.88×	6.41×10^{-3}	0.927	94.23	0.108
	SVD 4	15.94×	2.33×10^{-3}	0.377	96.38	0.0546
	SVD 8	7.97×	8.17×10^{-4}	0.129	98.03	0.0229
	SVD 16	3.98×	2.53×10^{-4}	0.0354	98.73	7.30×10^{-3}
	SVD 32	1.99×	7.52×10^{-5}	7.75×10^{-3}	99.23	1.57×10^{-3}
	INT8	4.00×	1.01×10^{-3}	0.177	97.66	4.99×10^{-3}
Full	SVD 1	63.75×	0.0107	1.29	93.64	0.200
	SVD 2	31.88×	4.86×10^{-3}	0.537	95.59	0.118
	SVD 4	15.94×	1.76×10^{-3}	0.275	97.14	0.0626
	SVD 8	7.97×	5.06×10^{-4}	0.103	98.19	0.0287
	SVD 16	3.98×	1.63×10^{-4}	0.0460	98.64	0.0106
	SVD 32	1.99×	5.59×10^{-5}	0.0142	99.00	2.91×10^{-3}
	INT8	4.00×	3.98×10^{-4}	0.129	97.80	4.97×10^{-3}

Table 7: Nemotron-3-Nano-4B: paired single-step prediction fidelity for 80×128 recurrent matrices. Storage is nominal reduction relative to FP32; INT8 uses the per-head affine baseline.

Prefix	Method	Storage	Median KL	p_{99} KL	Top-1 (%)	relFro^2
256	SVD 1	49.00×	0.0102	2.19	91.34	0.0397
	SVD 2	24.50×	3.55×10^{-3}	0.900	94.88	0.0151
	SVD 4	12.25×	1.11×10^{-3}	0.209	97.30	5.52×10^{-3}
	SVD 8	6.12×	2.87×10^{-4}	0.0326	98.64	1.75×10^{-3}
	SVD 16	3.06×	7.93×10^{-5}	6.40×10^{-3}	99.31	4.39×10^{-4}
	SVD 32	1.53×	2.99×10^{-5}	2.76×10^{-3}	99.42	7.28×10^{-5}
	INT8	4.00×	0.0216	7.32	85.50	1.70×10^{-3}
Full	SVD 1	49.00×	0.0106	1.32	96.30	0.0325
	SVD 2	24.50×	1.89×10^{-3}	0.381	97.88	0.0125
	SVD 4	12.25×	1.38×10^{-4}	0.107	98.72	4.72×10^{-3}
	SVD 8	6.12×	4.46×10^{-5}	0.0282	99.38	1.62×10^{-3}
	SVD 16	3.06×	1.39×10^{-5}	6.39×10^{-3}	99.61	4.61×10^{-4}
	SVD 32	1.53×	4.38×10^{-6}	2.16×10^{-3}	99.80	9.09×10^{-5}
	INT8	4.00×	0.0292	6.59	86.84	1.64×10^{-3}

Table 8: GatedDeltaNet-1.3B: paired single-step prediction fidelity for 256×256 recurrent matrices. Storage is nominal reduction relative to FP32; INT8 uses the per-head affine baseline.

Prefix	Method	Storage	Median KL	p_{99} KL	Top-1 (%)	relFro^2
256	SVD 1	127.75×	0.827	9.71	52.83	0.413
	SVD 2	63.88×	0.484	7.50	61.52	0.280
	SVD 4	31.94×	0.221	5.59	74.92	0.176
	SVD 8	15.97×	0.0605	3.11	87.73	0.0990
	SVD 16	7.98×	0.0103	1.07	95.00	0.0480
	SVD 32	3.99×	1.91×10^{-3}	0.203	97.81	0.0184
	INT8	4.00×	1.46×10^{-4}	4.18×10^{-3}	99.45	6.75×10^{-3}
Full	SVD 1	127.75×	1.23	11.1	47.83	0.426
	SVD 2	63.88×	0.958	9.55	53.66	0.299
	SVD 4	31.94×	0.553	7.77	63.02	0.199
	SVD 8	15.97×	0.204	5.43	77.02	0.125
	SVD 16	7.98×	0.0514	3.05	88.36	0.0729
	SVD 32	3.99×	0.0120	1.17	93.77	0.0374
	INT8	4.00×	2.15×10^{-4}	9.78×10^{-3}	99.45	6.45×10^{-3}

Table 9: Mamba2-1.3B: paired single-step prediction fidelity for 64×128 recurrent matrices. Storage is nominal reduction relative to FP32; INT8 uses the per-head affine baseline.

Prefix	Method	Storage	Median KL	p_{99} KL	Top-1 (%)	relFro^2
256	SVD 1	42.45×	0.153	6.08	75.16	0.139
	SVD 2	21.22×	0.0671	4.20	83.25	0.0687
	SVD 4	10.61×	0.0192	2.75	90.83	0.0291
	SVD 8	5.31×	3.77×10^{-3}	0.792	96.19	0.0104
	SVD 16	2.65×	6.82×10^{-4}	0.0679	98.39	2.84×10^{-3}
	SVD 32	1.33×	1.87×10^{-4}	0.0128	99.22	4.27×10^{-4}
	INT8	4.00×	4.09×10^{-4}	0.0230	98.95	1.40×10^{-3}
Full	SVD 1	42.45×	0.272	4.59	75.63	0.0872
	SVD 2	21.22×	0.0860	2.95	85.73	0.0406
	SVD 4	10.61×	0.0281	1.87	90.95	0.0175
	SVD 8	5.31×	8.83×10^{-3}	0.805	94.48	6.98×10^{-3}
	SVD 16	2.65×	3.46×10^{-3}	0.264	96.89	2.34×10^{-3}
	SVD 32	1.33×	1.38×10^{-3}	0.0688	98.13	4.80×10^{-4}
	INT8	4.00×	0.0377	1.70	91.14	1.53×10^{-3}

Table 10: DeltaNet-1.3B: paired single-step prediction fidelity for 128×128 recurrent matrices. Storage is nominal reduction relative to FP32; INT8 uses the per-head affine baseline.

Prefix	Method	Storage	Median KL	p_{99} KL	Top-1 (%)	relFro^2
256	SVD 1	63.75×	1.96	10.1	38.89	0.226
	SVD 2	31.88×	1.15	8.72	49.56	0.116
	SVD 4	15.94×	0.373	5.28	69.66	0.0510
	SVD 8	7.97×	0.0794	2.88	86.16	0.0213
	SVD 16	3.98×	0.0109	1.16	95.20	8.50×10^{-3}
	SVD 32	1.99×	8.58×10^{-4}	0.0800	98.88	2.47×10^{-3}
	INT8	4.00×	0.0153	1.31	94.00	7.04×10^{-3}
Full	SVD 1	63.75×	1.89	10.7	40.55	0.482
	SVD 2	31.88×	1.27	8.42	50.92	0.332
	SVD 4	15.94×	0.679	6.77	61.84	0.227
	SVD 8	7.97×	0.306	4.79	74.52	0.151
	SVD 16	3.98×	0.0920	2.86	86.30	0.0908
	SVD 32	1.99×	0.0156	0.841	94.13	0.0424
	INT8	4.00×	3.72×10^{-3}	0.402	96.52	6.35×10^{-3}

B Downstream Evaluation Details

Large-model periodic recompression. MMLU-Pro, GPQA-Diamond, and LongBench v2 use Qwen3.5-35B-A3B and NVIDIA-Nemotron-3-Nano-30B-A3B-BF16. For both models and all three benchmarks, compression uses rank-16 randomized SVD with four oversampling dimensions and one power iteration; recurrent states and SVD factors use FP32. We compress and reconstruct the recurrent states after prefill and every 1,024 generated tokens, leaving attention KV and convolution states unchanged. This tests repeated truncation during generation; GSM8K below applies compression only once.

Sampling parameters. For Qwen3.5-35B-A3B and Nemotron-3-Nano-30B-A3B, all three benchmarks (MMLU-Pro, GPQA-Diamond, and LongBench v2) use sampling with temperature 1.0, top-$p = 0.95$, top-$k = 20$, min-$p = 0$, presence penalty 1.5, and repetition penalty 1.0. Both models use a maximum generation length of 32,768 tokens. GPQA-Diamond uses seeds 42, 123, 456, and 789 for both models.

GPQA-Diamond repetitions. Table 3 reports the mean and sample standard deviation of the paired seed scores below. Entries are accuracies in percent.

	Qwen3.5-35B-A3B		Nemotron-3-Nano-30B-A3B	
Seed	Native	Rank 16	Native	Rank 16
42	82.32	80.30	67.68	66.67
123	81.82	81.82	63.64	69.70
456	80.30	81.31	62.63	62.63
789	82.83	81.31	68.18	64.65

We compute means and confidence intervals from the rounded per-seed accuracies above, using four paired seeds for each model. For each seed, we subtract native from compressed accuracy. The two-sided Student-t interval is $\bar{d} \pm t_{0.975,n-1} s_d/\sqrt{n}$, where $n = 4$ and s_d is the sample standard deviation of the paired differences. The resulting approximate 95% intervals are -0.63 $[-2.85, 1.58]$ percentage points for Qwen and $+0.38$ $[-6.09, 6.85]$ for Nemotron. These intervals describe seed variability; they do not quantify per-example uncertainty or establish noninferiority.

Other benchmark uncertainty. MMLU-Pro uses one run per condition; Table 3 reports the harness's per-example standard error. LongBench v2 also uses one run per condition, without confidence intervals; its score differences do not establish an accuracy improvement.

GSM8K one-time compression. The Qwen3.5-4B ablation uses all 1,319 GSM8K test examples through the LM Evaluation Harness (Gao et al., 2023), with five-shot prompting, batch size one, greedy decoding, and a 768-token generation cap. We compress the recurrent cache once between prefill and generation, then let decoding continue from the reconstructed state. Baseline states and SVD factors use FP32; randomized SVD uses four oversampling dimensions and one power iteration, matching the serving solver settings. Uniform quantization acts independently on each layer and head using simulated quantization and immediate dequantization rather than packed integer cache storage. These are simple state-level ablations, not implementations of Q-Mamba or Quamba2 (Chen et al., 2025; Chiang et al., 2025a).

C Scientific Ablations

Prompt length. Figure 4 shows that reconstruction error is nearly flat in prompt length except for DeltaNet. Mamba2 and GatedDeltaNet globally decay the prior state, while DeltaNet lacks this contraction; we hypothesize that this difference permits its effective rank to grow with context.

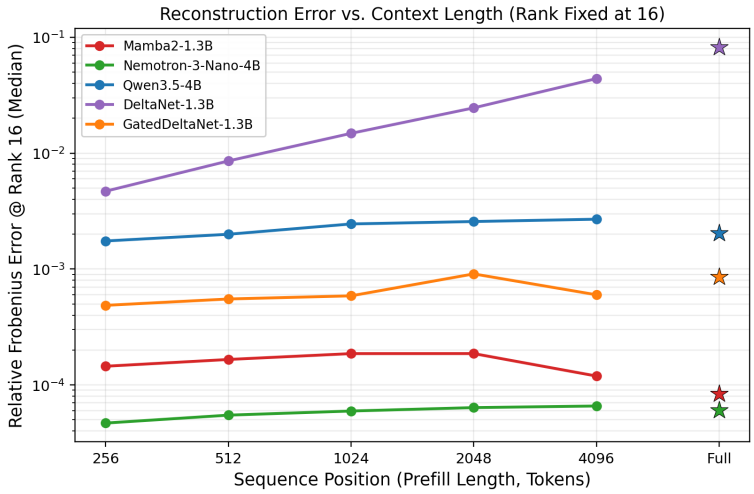

Figure 4: Median squared relative Frobenius error, $\|S - \hat{S}\|_F^2 / \|S\|_F^2$, at rank 16 on the 400-prompt characterization set. "Full" denotes each prompt's own full length, not one common token count. DeltaNet's error grows with context; the other four models remain nearly flat.

Rank and quantization. Table 11 compares recurrent-state compression on all 1,319 GSM8K examples. Rank-4 SVD has the same observed score as the baseline at $15.94\times$ nominal compression. Uniform 4-bit quantization loses 4.32 percentage points. SVD and quantization can also be combined by quantizing the stored low-rank factors, although we do not evaluate that combination here.

Table 11: Qwen3.5-4B GSM8K strict-extraction accuracy and nominal recurrent-state storage reduction relative to FP32. SVD ratios count stored FP32 factors; quantization ratios assume ideal bit packing without scale or rotation metadata.

Configuration	Method	Accuracy (%)	Compression
Uncompressed	None	84.99	1.00×
4-bit per head	Quantization	80.67	8.00×
8-bit per head	Quantization	85.44	4.00×
SVD rank 24	SVD	86.20	2.66×
SVD rank 16	SVD	85.22	3.98×
SVD rank 12	SVD	84.76	5.31×
SVD rank 8	SVD	86.28	7.97×
SVD rank 4	SVD	84.99	15.94×
SVD rank 2	SVD	82.79	31.88×
SVD rank 1	SVD	73.24	63.75×

D MATCHED-MEMORY SERVING PROTOCOLS

Hardware and software. The ShareGPT serving comparison uses Qwen3.5-4B on one NVIDIA GB10 (DGX Spark).[2]

Configurations. We pair compression disabled (off) with rank-16 compression enabled (on), using identical cache-memory ceilings, prompts, and target output lengths. Five paired repetitions replay recorded ShareGPT histories with eight active requests. Both modes use extra-buffer tracking with CUDA graphs and overlap scheduling disabled. Pool sizes are static.

Snapshot handoff and cache storage. The scheduler copies an eligible checkpoint into bounded staging storage on the scheduler stream, then records a readiness event. The SVD worker waits on this event before processing the snapshot on its dedicated CUDA stream. Completed factors enter a completion queue; the single-threaded scheduler installs the compressed-pool pointer before freeing the full-rank slot. Stale work for an evicted or detached radix node is discarded. This handoff preserves an immediately usable copy throughout compression and serializes metadata changes with ordinary cache operations. The compressed pool stores U_r, the singular values, and V_r contiguously for each layer and head, plus an exact copy of the convolution state. It has its own LRU list; evicting a compressed checkpoint may leave a recurrent-state tombstone while its attention KV blocks remain resident.

Worker scheduling. The asynchronous GPU worker processes at most two snapshots per SVD batch. It defers new batches while requests await prefill, selected or chunked prefill exists, or the prefill-completion event is unfinished. Already-admitted compression may overlap prefill; snapshot copies, restores, and commits are not deferred. Full-rank checkpoints remain authoritative until factors are installed. Staging capacity bounds outstanding snapshots independently of prefill deferral, which controls worker scheduling. Sustained prefill can starve compression.

ShareGPT trace construction. We use the unsplit, HTML-cleaned ShareGPT source, `anon8231489123/ShareGPT_Vicuna_unfiltered`, at revision `192ab2185289094fc556ec8ce5ce1e8e587154ca`. The two `sg_90k_part*_html_cleaned.json` files are concatenated without additional cleaning, language filtering, splitting, or deduplication. For each seed, we sample 64 conversations with at least ten human/assistant pairs, deterministically and without replacement within a run. Duplicate prompts across distinct records are retained. We replay the first ten pairs, excluding only malformed role/text sequences and empty tokenized assistant replies. Source URLs, checksums, sampled indices, source hashes, and exact token traces are saved with the protocol.

Reference-history replay. The native chat template is used with thinking disabled. Every later prompt contains the complete recorded conversation history; generated replies never replace that

[2]Model checkpoint: `851bf6e806efd8d0a36b00ddf55e13ccb7b8cd0a`.

history. Each response is decoded for the full token count of its recorded assistant reply with `ignore_eos`. There is no benchmark context cap, response-length cap, length-based filtering, or truncation. The server uses the model's native 262,144-token context without an override. Preflight fails if any selected request exceeds this limit, rather than dropping or shortening it. This protocol measures serving performance, not answer quality, and is not an exact reproduction of the Marconi traces.

The source provides no arrival timestamps or think times. All sessions are initially eligible in seeded order; each advances when its preceding request completes, sharing an eight-request semaphore. There are no global turn barriers or invented arrival timestamps. Actual dispatch times are logged and can differ between modes because completion times differ. The replay therefore preserves real conversation content and causality under closed-loop load, rather than reproducing the original production arrival process.

Tracking and pool occupancy. Both modes explicitly use `extra_buffer` recurrent-state scheduling and a 256-token tracking interval. Prefill tracks earlier aligned states; decoding tracks at 256-token intervals. Tracking buffers come from the fixed full-state pool. With overlap disabled, the scheduler reserves capacity at four full-state slots per admitted request, so 32 slots support eight active requests. These working and tracking reservations compete with retained checkpoints from successive conversation boundaries. A native chat template can change the generation-prompt suffix when a recorded reply becomes history, so the shared prefix may end before the previous full-prompt checkpoint. Extra-buffer tracking supplies earlier restore points; setting the interval alone does not enable this path in `no_buffer` mode.

Validation and measurement. Two pilots, each with eight conversations and ten turns, precede the five paired full measurements. Each must demonstrate at least 10% token hits and 50% of followup requests hitting cache. Startup verifies extra-buffer tracking, its interval, effective request concurrency, pool sizes, and staging budgets. Measured baseline runs must evict entries; otherwise the experiment stops without claiming pressure. Allocation, request-completion, compression-failure, and production-policy checks remain enforced. There are no automatic retries, resampling, or budget changes after failure. Mode order reverses on odd repetitions.

Memory accounting. Memory sizes use decimal MB (10^6 bytes) and GB (10^9 bytes). A full checkpoint occupies 51.5 MB and a compressed checkpoint 13.8 MB. Both include 1.18 MB of unchanged convolution state; recurrent matrices and SVD factors use FP32, independently of model-weight precision. BF16 checkpoint storage is not evaluated. Memory accounting includes reserved bookkeeping slots, called sentinels. Both modes fix KV capacity at 262,144 tokens (8.59 GB including its sentinel).

The 15.23 GB ceiling covers KV and state pools plus staging for snapshots and completed factors awaiting installation (Table 12). Compression-on persistent cache occupies 14.41 GB, leaving 0.83 GB below the ceiling for the 0.82 GB staging allowance. Bounded staging storage counts inside the ceiling. Arithmetic workspaces, including SVD scratch, model weights, allocator reserve, and metadata are outside this ceiling. We separately report peak process CUDA allocated memory, which is not the allocator's reserved-memory total.

Table 12: ShareGPT cache allocations under a common 15.23 GB ceiling. State slots exclude sentinels; accounted bytes include them. Staging is reserved within the ceiling.

Mode	Full slots	Compressed slots	Staging (GB)	KV tokens
Off	128	0	0	262144
On	32	298	0.82	262144

Cache metrics. Token hit rate is cached input tokens divided by total input tokens. State evictions count discarded checkpoints, not unique prefixes; the associated radix-tree nodes and attention KV can remain resident. KV-token evictions are counted separately. For ShareGPT, prior-input reuse opportunity is measured by the actual longest shared token prefix between inputs in the same session, accounting for chat-template suffix changes. Recomputed prior-input tokens count this opportunity

not served by the cache. Followup-hit fraction is the fraction of later-turn requests with a cache hit. These metrics estimate missed reuse from server metadata rather than directly measuring GPU work. Outstanding compression jobs include queued and executing tasks. Completion fractions are measured at the end of the run without waiting for outstanding jobs to finish.

Latency and throughput. Time to first token (TTFT) runs from request submission to the first streamed output-token event. It excludes time waiting at the client for a concurrency slot, but includes waiting inside the server. Time per output token (TPOT) is the interval from the first to last output-token event divided by output length minus one, with zero assigned to single-token replies. Throughput is completed requests or output tokens divided by wall time, including metric collection.

Statistics. Paired observations are on-minus-off differences across repetitions. Tail intervals summarize per-run percentiles, not pooled-request quantiles. Percentage changes use ratios of mode means. The ShareGPT comparison reports run means and paired 95% CIs over five repetitions. Each interval is a two-sided Student-t interval over paired run differences, $\bar{d} \pm t_{0.975,4} s_d/\sqrt{5}$, where s_d is the sample standard deviation of the five on-minus-off differences. Off/on order alternates; five repetitions leave a 3:2 split in which mode runs first. Pilot runs are excluded from these summaries. All repetitions share one GPU; temperature, clocks, and other processes are not fully controlled despite alternating execution order.

E Serving Results and Diagnostics

E.1 Multi-Turn ShareGPT Serving

Table 13 supplements Table 4 with paired confidence intervals, retention, and memory measurements. The comparison uses prefill deferral and a two-snapshot batch cap together; it does not isolate their individual contributions.

Table 13: ShareGPT reference-history replay at 15.23 GB and concurrency eight: 64 sessions with ten turns each. Means and paired on-minus-off 95% CIs over five repetitions. Differences use the supplied paired aggregates and can differ from subtraction of rounded mode means.

Metric	Off	On	Change [95% CI]
Token hit rate (%)	3.28	67.15	63.87 [62.65, 65.08]
Output throughput (tokens/s)	128.07	142.75	14.68 [13.80, 15.56]
Mean TTFT (ms)	482.5	251.5	−231.0 [−253.4, −208.7]
P50 TTFT (ms)	396.2	206.1	−190.1 [−219.8, −160.3]
P95 TTFT (ms)	1073.2	506.8	−566.4 [−684.8, −448.0]
P99 TTFT (ms)	1686.3	871.3	−815.0 [−1369.3, −260.7]
Mean TPOT (ms)	59.56	54.08	−5.47 [−6.05, −4.89]
Run duration (s)	1374.34	1232.98	−141.35 [−154.82, −127.88]
Request throughput (requests/s)	0.4667	0.5202	0.0535 [0.0490, 0.0580]
Evicted state entries	1191.6	793.6	−398.0 [−421.2, −374.8]
Evicted KV tokens	995406	205433	−789974 [−862264, −717683]
Recomputed prior-input tokens	831698	142294	−689404 [−740435, −638373]
Persistent cache (GB)	15.23	14.41	−0.83 [−0.83, −0.83]
Peak cache-state (GB)	15.23	14.85	−0.38 [−0.39, −0.38]
Peak staging (GB)	0.00	0.45	0.45 [0.44, 0.45]
Peak process CUDA (GB)	25.21	25.21	−0.0004 [−0.086, 0.085]

Reuse opportunity and retention. The paired traces contain a mean of 866,903 prior-input prefix tokens, representing 80.30% of input tokens by the reported per-run fraction. The fraction of followup requests hitting cache rises from 4.03% to 81.60%, a paired increase of 77.57 percentage points (95% CI [75.90, 79.24]). All compression-on state evictions are compressed entries; neither mode reports retractions.

Worker progress and accounting. Compression enqueues a mean of 1,092.6 checkpoints and completes and installs 1,091.6 per run, with zero failures. The mean completion fraction is 99.9084%

at measurement end, with a mean of 470.2 decompression hits per run. Outstanding jobs peak at a mean of five, and the worker completes 0.887 checkpoints/s. Mean SVD batch size is 1.012, with a maximum of two. Prefill admission defers 383.4 batches per run, with 11.71 s of aggregate deferral waiting; this is worker waiting time, not added request latency. Process-allocation differences measure net process peaks rather than isolated SVD workspace.

E.2 Trace-Driven Simulation Protocol

The Marconi simulation (Pan et al., 2025) measures cache reuse, not wall-clock latency. It models fourfold recurrent-matrix compression for Qwen3.5-4B with unchanged KV and convolution bytes, distinct from the $3.73\times$ full-checkpoint reduction in live serving.

Workload grid. Session arrival rates are 0.25, 0.5, 1, 2, 5, and 10 sessions/s. Cache budgets are 1–5 GB for LMSYS-Chat-1M, 1, 2, 3, 4, 5, and 10 GB for ShareGPT, and 40, 60, 80, and 100 GB for SWE-bench. SWE-bench additionally varies the average response time between requests within a session: 5, 7.5, or 10 seconds. The resulting grids contain $5 \times 6 = 30$ LMSYS settings, $6 \times 6 = 36$ ShareGPT settings, and $4 \times 6 \times 3 = 72$ SWE-bench settings per policy.

Aggregation and policy selection. For each policy family and grid setting, Table 5 subtracts the uncompressed token hit rate from the compressed token hit rate, then averages these differences with equal weight across settings. Individual hit rates are token-weighted within a trace; the grid average is not weighted by request count, token count, budget, or arrival rate. For SWE-bench, each budget/rate/response-time combination receives equal weight.

Policy definitions. Admit-all checkpoints every block and uses block LRU. Selective retains sequence-end checkpoints and checkpoints at shared-prefix branch points rather than at every block boundary, with LRU eviction among leaves. Marconi uses selective admission and online tuning of an eviction score combining recency with FLOPs saved per stored byte. Best-static selects the fixed weight α that maximizes token hit rate over candidate values offline, separately for compressed and uncompressed conditions. The weight controls FLOP efficiency relative to recency in the eviction score. The comparison therefore measures compression with each condition independently tuned. An online policy can vary this weight over time and outperform the best fixed value; Best-static is not an upper bound on adaptive policies.

F Limitations and Future Work

Live-serving gains are measured on one GB10, one cache budget, and closed-loop ShareGPT replay with recorded histories. Generalization to model-generated histories, production arrivals, multiple devices, or tenants remains untested. Graphs and overlap scheduling are disabled, and the matched cache ceiling excludes model weights and runtime workspaces.

The quality evaluations do not establish statistical equivalence or quality under repeated cache save-and-restore cycles in live serving. Adaptive ranks and full-rank fallbacks may address the observed reconstruction-error tails. Dynamic pool resizing could reclaim unused capacity while reserving working and staging space, but its retention and latency benefits require evaluation.

G Broader Impacts

Smaller caches could lower serving costs and broaden access to capable models, including their harmful uses; energy savings are not measured. Approximate restoration can alter individual generations despite stable aggregate scores, so sensitive deployments should monitor task quality and retain full-rank fallbacks.