DASC: DECAY-AWARE STATE COMPRESSION FOR HYBRID LINEAR-ATTENTION SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid linear-attention architectures have recently scaled to large open-weight models, offering quality competitive with full attention while substantially reducing key/value (KV) cache growth. However, their in-place recurrent-state updates complicate cache management: prefix reuse requires state checkpoints alongside full-attention KV, while storing state checkpoints in full increases memory pressure, leading to more evictions and repeated prefill. By analyzing the decay structure of Gated DeltaNet (GDN) and Kimi Delta Attention (KDA), we find that different heads and channels retain prefix information over markedly different timescales, which we term *retention horizons*. This variation suggests substantial compression potential in persistent state checkpoints. Building on this observation, we introduce *Decay-Aware State Compression* (DASC), which derives retention horizons from model weights, packs selected long-horizon units into ragged state checkpoints, and optionally balances their storage across tensor-parallel ranks. On reuse, DASC either zero-fills omitted units or refreshes them from a bounded suffix with additional compute cost. We evaluate DASC on Kimi-Linear, Qwen3-Next, and Kimi-K3, covering model quality and serving performance. Across the evaluated retrieval and reasoning tasks, conservative configurations maintain quality close to full state checkpoint caching, while suffix refresh recovers accuracy at more aggressive compression. Implemented in SGLang, DASC provides 1.32–$2.22\times$ state checkpoint compression at representative operating points, reducing mean Time to First Token (TTFT) by 16.1–42.0% and mean end-to-end (E2E) latency by 4.9–10.2%.

1 INTRODUCTION

Hybrid linear-attention architectures now underpin trillion-parameter open-weight models, including Qwen3.8-2.4T-A95B and the 2.8T-parameter Kimi-K3 (Qwen Team, 2026; Kimi Team et al., 2026). These models interleave full-attention layers with linear-attention layers that summarize the prefix in fixed-size recurrent states. This hybrid design retains strong model quality while slowing key/value (KV) cache growth. However, its in-place recurrent-state updates introduce a new challenge for prefix caching.

Unlike full-attention KV, a recurrent state is overwritten as tokens arrive and does not preserve earlier prefix boundaries. Hybrid prefix caching therefore stores recurrent state checkpoints alongside full-attention KV at reusable prefix boundaries (Pan et al., 2025; Kimi Team et al., 2026). Because each state checkpoint contains large state matrices for all linear-attention layers, frequent state checkpointing quickly exhausts HBM, while sparse state checkpointing increases replay or repeated prefill. This tension makes it important to determine which state units must be retained at each state checkpoint.

Our analysis reveals a structured answer: recurrent-state units have widely different retention horizons. Many forget distant tokens quickly, while a smaller subset remains persistent. This variation follows the model architecture, appearing across heads in Gated DeltaNet (GDN) and across channels in Kimi Delta Attention (KDA). Moreover, the decay parameters stored in model weights provide an input-independent estimate of these horizons; Figure 2 illustrates how they drive state checkpoint selection and reuse. We validate this signal against observed token-dependent decay and state magnitudes. Controlled cache ablations further show why selection must be conservative:

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.

recurrent states can be redundant for simple recall yet remain important for multi-key, aggregation, and reasoning tasks.

We translate these findings into *Decay-Aware State Compression* (DASC). Before serving, DASC constructs a static selection plan that retains long-horizon heads for GDN or channels for KDA. At runtime, selected units are flat-packed into ragged state checkpoints with optional tensor-parallel placement balancing; full-attention KV remains unchanged. In SGLang, a two-tier cache combines compressed storage with native slots shared by running states and full cached state checkpoints. On a compressed state checkpoint hit, DASC-NR zero-fills omitted units without additional model computation, whereas DASC-WR refreshes them from a bounded suffix at additional compute cost.

We evaluate DASC on Kimi-Linear-48B-A3B-Instruct (Kimi Team et al., 2025), Qwen3-Next (Qwen Team, 2025), and Kimi-K3 (Kimi Team et al., 2026) using retrieval and reasoning tasks together with natural WildChat-1M serving traces (Zhao et al., 2024). Conservative configurations remain close to dense state checkpoint quality, while suffix refresh recovers accuracy at more aggressive compression. Representative operating points reduce mean TTFT by 42.0%, 25.6%, and 16.1% and mean E2E latency by 10.2%, 4.9%, and 6.0% on the three models, respectively.

Our contributions are:

- We identify the storage pressure of full state checkpoints in hybrid-model prefix caching and characterize decay heterogeneity in GDN and KDA, deriving weight-based retention horizons and validating their ordering against token-dependent decay across three models.
- We design and implement DASC in SGLang, integrating architecture-aware selection, ragged storage, optional TP-balanced placement, and two-tier state checkpoint management. Compressed state checkpoints use zero-fill by default; optional bounded-suffix refresh provides accuracy recovery at additional compute cost.
- Across Kimi-Linear, Qwen3-Next, and Kimi-K3, conservative configurations maintain quality close to full state checkpoints. Representative operating points achieve 1.32–$2.22\times$ state checkpoint compression, reducing mean TTFT by 16.1–42.0% and mean E2E latency by 4.9–10.2%.

2 Related Work

Linear attention and hybrid architectures. Linear attention and state-space models replace the context-length-dependent KV cache of softmax attention with a fixed-size recurrent state (Katharopoulos et al., 2020; Gu & Dao, 2024; Dao & Gu, 2024). Delta-rule and gated variants add content-dependent updates and adaptive decay (Yang et al., 2024; 2025), while hybrid models interleave recurrent and full-attention layers to preserve direct access to token-level KV representations (Lieber et al., 2024; Ren et al., 2025; Kimi Team et al., 2025). Unlike architecture and training work, we study how the resulting recurrent states should be represented in a serving-time prefix cache.

Prefix caching and full-attention KV management. SGLang's RadixAttention organizes shared prefixes in a radix tree, while vLLM's PagedAttention provides block-grained KV allocation and reuse (Zheng et al., 2024; Kwon et al., 2023). Hierarchical systems extend reusable KV across GPU and host memory (Gao et al., 2024; Jin et al., 2025); quantization, eviction, and sparsification further reduce token-indexed KV storage (Hooper et al., 2024; Liu et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024). These methods are complementary to DASC, which leaves full-attention KV unchanged and compresses the recurrent state checkpoint stored alongside it.

Hybrid recurrent-state prefix caching and replay. Processing a prefix through a recurrent layer produces one boundary state summarizing all preceding tokens, so reuse requires storing that state or recomputing it. Kimi-K3 persists KDA states for prefix reuse (Kimi Team et al., 2026); Marconi instead improves which hybrid prefix entries are admitted and evicted based on reuse and compute–memory utility (Pan et al., 2025). ReplaySSM caches recent SSM inputs and reconstructs states on demand, targeting decode-time state traffic and rollback (Liou & Dao, 2026). Q-Mamba quantizes recurrent state caches alongside weights and activations (Tianqi et al., 2025). Whereas quantization reduces bits per value, DASC selects which units to store, complementing both precision reduction and entry-management policies.

Table 1: Evidence for recurrent-state compression opportunity. **(a)** Cached-state intervention on RULER NIAH-S1. **(b)** Within-layer Spearman correlations: H_s is the static horizon, H_e the empirical horizon, and M_S the state norm. Values are median [IQR].

(a) Cached-state intervention

Intervention	Qwen3-Next	Kimi-Linear	Observation
BASELINE	1.000	1.000	reference
ZERO-FULL-KV	0.000	0.000	retrieval fails
ZERO-LINEAR	0.990	1.000	$\leq$ 1-point change
ZERO-BOTH	0.000	0.000	sanity check

(b) Static horizons vs. observed signals

Model	Pair	Median	IQR
Qwen3-Next	H_s, H_e	.801	[.699, .877]
Qwen3-Next	H_s, M_S	.632	[.541, .706]
Kimi-Linear	H_s, H_e	.856	[.843, .878]
Kimi-Linear	H_s, M_S	.533	[.462, .602]
Kimi-K3	H_s, H_e	.895	[.867, .941]
Kimi-K3	H_s, M_S	.443	[.363, .513]

3 Retention Heterogeneity in Hybrid Recurrent States

This section first uses a controlled cache ablation to identify task-specific redundancy in recurrent state checkpoints. The decay structures of Qwen3-Next's GDN and the KDA modules in Kimi-Linear and Kimi-K3 then motivate a static, serving-input-independent metric for estimating retention horizons. Empirical validation supports its use as a ranking heuristic for state selection, providing the basis for DASC.

3.1 Recurrent states in hybrid prefix caches

A gated delta-rule layer summarizes the processed sequence in a recurrent state $S_t \in \mathbb{R}^{d_v \times d_k}$, which a query reads as $o_t = S_t q_t$. KDA updates this state as

$$S_t \;=\; S_{t-1}\,\mathrm{Diag}(\alpha_t)\left(I - \beta_t k_t k_t^\top\right) \;+\; \beta_t\, v_t k_t^\top, \qquad \alpha_t \in (0,1)^{d_k}, \tag{1}$$

where α_t controls input-dependent forgetting and β_t the update strength (Kimi Team et al., 2025). The three operations attenuate the old state, remove content aligned with k_t, and write the new key–value association. GDN applies gated decay per head (Yang et al., 2025), whereas KDA exposes channel-wise decay within each head.

Unlike full attention, a delta-rule layer carries only this fixed-size state forward. Reusing prefix p therefore requires both exact full-attention KV and each recurrent layer's boundary state $S^{(l)}(p)$. In prefix caching, each reusable prefix requires its own *recurrent state checkpoint*; if the state checkpoint is not cached, the state must be recomputed through prefill.

3.2 Recurrent state checkpoints contain task-specific redundancy

We probe these two cache components on RULER NIAH-S1 using five needle depths and $N = 100$ examples per depth. For Qwen3-Next-80B-A3B-Instruct and Kimi-Linear-48B-A3B-Instruct, we cache the shared prefix and zero its full-attention KV, recurrent state checkpoints, both, or neither before reuse. Every intervention reuses the same fully cached prefix, so any accuracy difference comes from the zeroed cache component rather than from differences in cache hits.

Removing full-attention KV reduces accuracy to zero, whereas removing recurrent state checkpoints changes it by at most one point (Table 1(a)). This probe is consistent with retrieval being carried by full-attention KV despite recurrent-state removal. It motivates testing compression, but does not establish general state redundancy; harder tasks in §5.2 assess its scope.

3.3 Static horizons enable input-independent selection

The intervention suggests task-specific tolerance to state omission, but does not identify which units to retain. Although the input-dependent decay gate provides a natural retention signal, collecting it for every prompt would make the selection request-dependent. An offline serving plan instead requires a fixed metric that does not depend on serving requests.

To build this metric, DASC evaluates each architecture's static log-decay. Qwen3-Next's GDN and Kimi-Linear's KDA use the standard softplus gate, while Kimi-K3's KDA uses a bounded safe

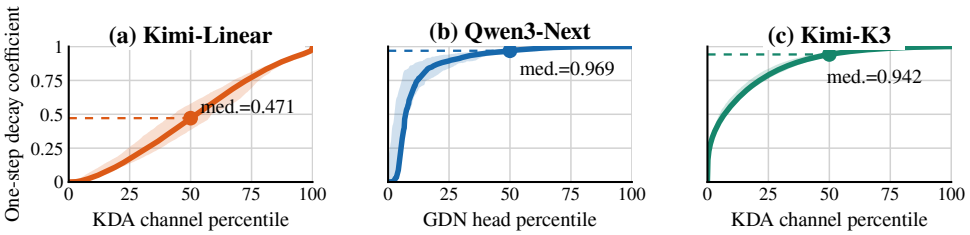


Figure 1: Weight-only one-step decay profiles for Kimi-Linear KDA, Qwen3-Next GDN, and Kimi-K3 KDA.

gate (Kimi Team et al., 2025; 2026):

$$\begin{aligned} g_s^{\mathrm{std}}(u) &= -\exp(A_{\log,u})\,\mathrm{softplus}(a_s + d_u), \\ g_s^{\mathrm{K3}}(u) &= \ell\,\sigma(\exp(A_{\log,u})(a_s + d_u))\,, \quad \ell = -5. \end{aligned} \tag{2}$$

where $A_{\log,u}$ is a learned log-scale parameter, d_u the learned decay bias, and $\sigma(x) = (1 + e^{-x})^{-1}$ the sigmoid function. The constant $\ell = -5$ is Kimi-K3's safe-gate lower bound. By contrast, DASC replaces the input-dependent gate input $a_{t,u}$ with a fixed representative value a_s, shared across all recurrent layers and units of a model: $a_s = -0.3$ for Qwen3-Next and Kimi-Linear, and $a_s = 0.8$ for Kimi-K3. With $\alpha_s = \exp(g_s)$ and $g_s < 0$, we define the *static retention horizon* as

$$H_s = \frac{\ln \epsilon}{g_s}, \qquad \epsilon = 10^{-3}. \tag{3}$$

For Qwen3-Next and Kimi-Linear, setting $a_s = -0.3$ yields slower static decay and longer retention horizons than $a_s = 0$, biasing the selector toward retaining more units. For Kimi-K3, DASC instead evaluates its native bounded safe gate using the calibrated representative $a_s = 0.8$.

Figure 1 profiles static one-step decay across the three models. Kimi-Linear exhibits broad channel-wise decay (median $\alpha = 0.471$), whereas Qwen3-Next GDN heads and Kimi-K3 KDA channels are predominantly slow-decaying (medians 0.969 and 0.942). GDN assigns one horizon per head, while KDA resolves differences among channels within each head. At a 16-token cutoff, 60.8% of Kimi-Linear KDA layer–head pairs contain both retained and omitted channels; at a 64-token cutoff, the corresponding fraction is 98.3% for Kimi-K3. This within-head variation motivates channel-wise KDA selection. Across layers, unequal retained-unit counts motivate ragged storage; Appendix Figures 5–8 show the model-specific structure and statistics. These weight-derived, request-independent profiles characterize storage heterogeneity, not the importance of omitted units.

3.4 Weight-derived horizons track observed decay

We assess how static horizons correlate with empirical horizons and observed state magnitudes across all three models. Qwen3-Next and Kimi-Linear use 32 length-stratified ShareGPT prefixes, while Kimi-K3 uses 16 held-out prefixes spanning 128–2,048 tokens. Static horizons follow the model-specific gate parameterizations and representative values defined in Section 3.3. Appendix A.2 details the protocols.

For prefix p, recurrent layer l, and state unit u (a GDN head or a KDA key channel), let $\mathcal{T}_p$ denote the non-padding token positions and $g_{p,t,l,u}$ the observed log-decay. The *empirical horizon* is

$$\bar{g}^{(e)}_{p,l,u} = \frac{1}{|\mathcal{T}_p|} \sum_{t \in \mathcal{T}_p} g_{p,t,l,u}, \qquad H^{(e)}_{p,l,u} = \frac{\ln \epsilon}{\bar{g}^{(e)}_{p,l,u}}. \tag{4}$$

We also measure state magnitude at the final prefix position. Suppressing prefix, layer, and time indices, let S_h denote a head state and $S_{h,k}$ a state channel:

$$M_S(u) = \begin{cases} \|S_h\|_F, & u = h \text{ (GDN)}, \\ \|S_{h,k}\|_2, & u = (h,k) \text{ (KDA)}. \end{cases} \tag{5}$$

For each prefix–layer pair, we compute Spearman correlations across units between H_s and each of H_e and M_S. Table 1(b) reports medians and IQRs. The median horizon correlations are 0.801,

0.856, and 0.895 for Qwen3-Next, Kimi-Linear, and Kimi-K3; the corresponding state-magnitude correlations are 0.632, 0.533, and 0.443. These results support static horizons as a ranking heuristic associated with observed state activity, rather than a complete importance measure. Rank agreement does not establish agreement at a fixed horizon cutoff; the downstream quality experiments evaluate the effect of the resulting state selection.

4 DECAY-AWARE STATE COMPRESSION

Using the static horizons H_s from §3, DASC uses an input-independent mask to retain long-horizon state units and optionally refreshes omitted units on reuse. Figure 2 summarizes the store and load paths.

4.1 DECAY-AWARE SELECTION AND STORAGE

DASC constructs a fixed selection mask using a retention-horizon cutoff $W_{\max}$, measured in tokens. A unit is *global* and retained when $H_s(u) > W_{\max}$; otherwise, it is *local* and omitted from the persistent state checkpoint. Selection follows the architecture's decay granularity: one complete head state for GDN and one key channel (h, k), corresponding to $S^{(l)}_{h,:,k}$, for KDA. After fixing the model-level representative a_s, the mask is constructed once offline and remains independent of serving requests.

Let L be the number of recurrent linear-attention layers, excluding full-attention layers, and let n_l and r_l be the total and retained state-unit counts in layer l, respectively. A unit is one head in GDN and one (head, k) channel in KDA. Units have equal cost within the temporal-state tensor, so ragged storage keeps $\sum_l r_l$ temporal units instead of $\sum_l n_l$. Recurrent state checkpoints use SGLang's default mixed precision: the temporal state is FP32 and the convolution state is BF16; BF16 model execution is a separate setting. Let B_{temp} and B_{conv} denote their physical bytes. The temporal-only and full state checkpoint compression ratios are

$$C^{\text{rag}}_{\text{temp}} = \frac{\sum_{l=1}^{L} n_l}{\sum_{l=1}^{L} r_l}, \qquad C^{\text{rag}}_{\text{ckpt}} = \frac{B^{\text{dense}}_{\text{temp}} + B_{\text{conv}}}{B^{\text{rag}}_{\text{temp}} + B_{\text{conv}}}. \tag{6}$$

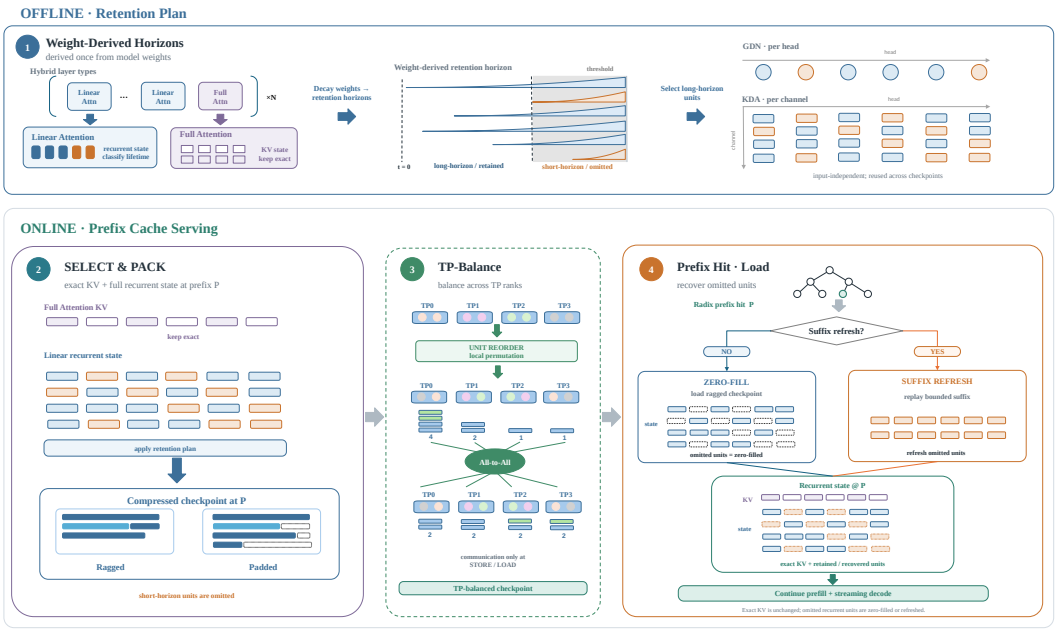


Figure 2: **Overview of DASC.** Weight-derived retention horizons select long-horizon state units, which are packed into ragged state checkpoints with optional TP-balanced placement. On reuse, omitted units are either zero-filled or refreshed from a bounded suffix.

At fixed state checkpoint memory, C_{ckpt} is the ideal capacity multiplier, before slot rounding and allocator reservations; actual capacity also depends on per-rank allocation (Appendix C). A **padded** temporal layout reserves the largest retained count $R_{\max} = \max_l r_l$ for every layer, even though only r_l entries are valid in layer l, giving $C_{\text{temp}}^{\text{pad}} = (\sum_l n_l)/(LR_{\max})$. The **ragged** layout flat-packs each layer's retained units with offsets. With $\bar{r} = L^{-1}\sum_l r_l$, its layout gain over padded storage is $R_{\max}/\bar{r}$. Reported capacity ratios include the uncompressed BF16 convolution state and the per-rank TP bottleneck (Appendix B); the gain is large for KDA's imbalanced retained counts and modest for GDN's more uniform profile.

4.2 Load policies

On a compressed state checkpoint hit, DASC loads the retained global units exactly. **DASC-NR** (the default) leaves omitted local units at zero. **DASC-WR** instead replays a bounded suffix from a zero-initialized scratch recurrent state and copies only the refreshed local units into the active state; the stored global state checkpoint and cached full-attention KV remain unchanged. Let $\mathcal{L}_{\text{local}}$ denote the omitted units and n_p the cached prefix length. For $W_{\max} \geq 4$, define $\text{next_pow2}(x) = 2^{\lceil \log_2 x \rceil}$ for $x > 0$, mapping zero to 4. The longest per-unit window is capped by the available prefix; if no units are omitted, $W_{\text{replay}} = 0$ and refresh is skipped. Otherwise,

$$W_u = \text{clamp}(\text{next_pow2}(H_s(u)), 4, W_{\max}) \qquad W_{\text{replay}} = \min\left(n_p, \max_{u \in \mathcal{L}_{\text{local}}} W_u\right). \tag{7}$$

Increasing $W_{\max}$ omits more units and improves state checkpoint compression, but can increase DASC-NR approximation error and DASC-WR replay cost. During scratch replay, all recurrent units, including global units, start from zero. Missing pre-window history therefore affects intermediate activations across layers, not only the local-state initial condition. Only refreshed local units are copied back; retained active global units remain exact. Thus DASC-WR is approximate recovery, not an exact reproduction of dense prefix computation.

4.3 TP-balanced placement

With uneven rank payloads U_r, TP state checkpoint capacity is limited by $\max_r U_r$. DASC optionally permutes head bundles and all coupled tensors at model loading. Independently, storage balancing redistributes packed units across T ranks with target slot width $C = \lceil \sum_r U_r/T \rceil$. A variable-split all-to-all transfers only remotely owned units during STORE; LOAD reverses the mapping to restore the compute-local layout. Both stages are exact reindexings; communication occurs only at state checkpoint boundaries, while recurrence remains rank-local. Storage balancing also works without permutation. Appendix B.1 gives the assignment algorithms and compatibility constraints.

4.4 SGLang integration

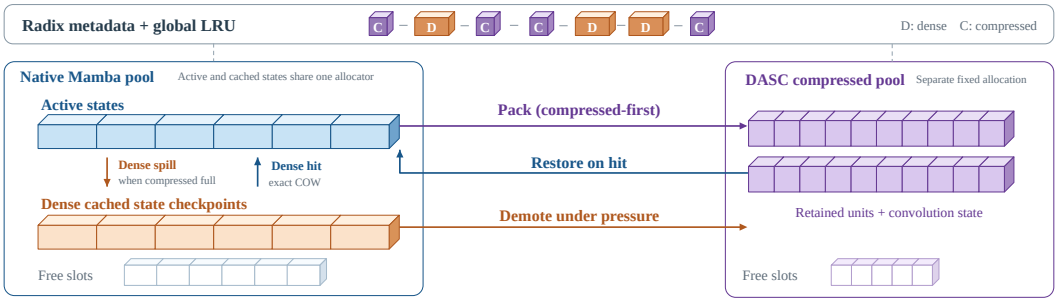


Figure 3: Two-tier state checkpoint cache in SGLang.

Figure 3 illustrates DASC's two-tier state checkpoint cache in SGLang. The native Mamba pool shares a slot allocator between running states and full cached state checkpoints, while a separately allocated compressed pool stores retained recurrent units and uncompressed convolution states. Radix metadata tracks each cached prefix's tier, slot, and lock count, and a global LRU maintains a single

recency order across both cached representations. KV allocation remains separate. Appendix B.2 details insertion, cache reuse, eviction, and asynchronous demotion.

5 EXPERIMENTS

We evaluate DASC for quality, matched-HBM WildChat serving, and the accuracy–latency trade-off of suffix refresh.

5.1 SETUP

Models and hardware. We evaluate Qwen3-Next-80B-A3B-Instruct, which contains 36 GDN and 12 full-attention layers (Qwen Team, 2025), and Kimi-Linear-48B-A3B-Instruct, which contains 20 KDA and 7 full-attention layers (Kimi Team et al., 2025). The additional serving and five-task quality study includes Kimi-K3 (Kimi Team et al., 2026). We use Kimi-Linear, Qwen3-Next, and Kimi-K3 for model-level results, and GDN/KDA only for their recurrent architectures. Qwen3-Next exposes head-wise GDN decay, whereas the two Kimi models expose channel-wise KDA decay. Kimi-Linear and Qwen3-Next run on eight Hopper-architecture GPUs with TP8.

Benchmarks. Long-context quality is evaluated on all 13 RULER subtasks at 4k, 8k, and 16k context lengths, with $N = 30$ instances per subtask–length setting (Hsieh et al., 2024). The end-to-end suite covers mathematical reasoning with the MathArena releases of AIME 2026 Parts I and II and HMMT February 2026 (Dekoninck et al., 2026), together with IMO-AnswerBench (Luong et al., 2025); general reasoning with GPQA-Diamond (Rein et al., 2024) and MMLU-Pro (Wang et al., 2024); and conversational memory with LoCoMo (Maharana et al., 2024). Serving performance uses natural conversations from WildChat-1M (Zhao et al., 2024).

Implementation and evaluation protocol. We implement DASC in SGLang (Zheng et al., 2024). For Kimi-Linear and Qwen3-Next, model weights and execution are unquantized BF16. Dense denotes the baseline that caches full recurrent state checkpoints without DASC compression, using FP32 temporal states and BF16 convolution states. Unless noted, DASC-NR and ragged storage are used, and compression is relative to the dense mixed-precision state checkpoint, excluding unchanged full-attention KV. Table 2 and RULER use a strict warmup–replay protocol: an unscored warm request populates each prefix state checkpoint, and only replay outputs and pooled replay token-hit rates are reported. Replays are averaged within each unique problem or instance, which remains the statistical unit; paired arms share prompts, cache-population order, seeds, and slot counts. DASC and Random accuracy runs cache state checkpoints only in the compressed pool, with dense spill disabled; native dense slots serve active computation only. WildChat serving instead enables the two-tier cache with dense spill (Appendix B.2) and reports all-request means over a continuous trace. Appendix C gives task-specific counts and controls.

5.2 QUALITY UNDER RECURRENT-STATE COMPRESSION

RULER uses three measured replay rounds for each of $N = 30$ unique instances per subtask–length–arm setting. The NIAH-S1 diagnostic is saturated: dense and DASC-NR both score 1.000 for both models at every tested $W_{\max}$. We therefore report the broader RULER retrieval, extraction, QA, and tracking suite. Appendix Tables 5 and 6 report the complete Kimi-Linear and Qwen3-Next sweeps.

At $W_{\max} = 16$, Kimi-Linear scores $0.95/0.95/0.95$ at 4k/8k/16k versus dense $0.95/0.96/0.95$; Qwen3-Next scores $0.84/0.83/0.81$ versus $0.84/0.83/0.80$. Thus every displayed aggregate is within 0.01 of dense. At $W_{\max} = 1024$, Kimi-Linear drops by $0.04/0.06/0.07$, with the largest 16k losses on MK1, MV, CWE, and QA-S. Qwen3-Next aggregates remain within 0.01, although MK3 falls from 1.00 to 0.87 at 16k. Section 5.4 evaluates suffix refresh.

End-task quality. Table 2 applies the shared warmup–replay protocol to dense state checkpoints, ragged DASC-NR, and count-matched Random. Scores average measured replay generations within each problem, and Hit pools cached and new tokens across all compared arms in each row; arm-specific hit rates are similar. For each layer and $W_{\max}$, each of five fixed Random masks uni-

Table 2: Replay-only end-task quality. DASC entries are point estimates; Random is mean±sample SD across the same five fixed-mask seeds for all three models. LoCoMo reports token F1; other rows report accuracy. Hit is the replay token-hit rate pooled across all compared arms in each row.

Model	Benchmark	Hit	Dense	Ragged DASC-NR, by $W_{\max}$					
				$W_{\max}=16$		$W_{\max}=64$		$W_{\max}=128$	
				DASC	Random	DASC	Random	DASC	Random
Kimi-Linear	AIME 2026	0.762	0.6065	0.6032	0.5634 ± 0.0356	0.5398	0.5153 ± 0.0329	0.5699	0.4744 ± 0.0252
	HMMT 2026 Feb	0.705	0.3754	0.3861	0.3423 ± 0.0160	0.3343	0.3062 ± 0.0167	0.3099	0.2741 ± 0.0180
	IMOAnswerBench	0.546	0.2783	0.2675	0.2663 ± 0.0063	0.2717	0.2568 ± 0.0066	0.2642	0.2455 ± 0.0066
	GPQA-Diamond	0.716	0.6185	0.6131	0.5749 ± 0.0189	0.5579	0.5422 ± 0.0083	0.5704	0.5309 ± 0.0111
	MMLU-Pro	0.752	0.6750	0.6732	0.6550 ± 0.0046	0.6458	0.6343 ± 0.0015	0.6386	0.6236 ± 0.0028
	LoCoMo (F1)	0.993	0.4995	0.4986	0.4883 ± 0.0093	0.4899	0.4640 ± 0.0070	0.4824	0.4568 ± 0.0141
Qwen3-Next	AIME 2026	0.621	0.6763	0.6871	0.6735 ± 0.0157	0.6763	0.6813 ± 0.0173	0.6753	0.6802 ± 0.0095
	HMMT 2026 Feb	0.578	0.3881	0.3861	0.3867 ± 0.0072	0.3803	0.3975 ± 0.0167	0.3949	0.3871 ± 0.0078
	IMOAnswerBench	0.499	0.3375	0.3400	0.3440 ± 0.0090	0.3383	0.3472 ± 0.0125	0.3417	0.3425 ± 0.0072
	GPQA-Diamond	0.819	0.6960	0.7013	0.7013 ± 0.0089	0.7051	0.6998 ± 0.0064	0.7044	0.7034 ± 0.0047
	MMLU-Pro	0.868	0.7875	0.7882	0.7857 ± 0.0082	0.7863	0.7828 ± 0.0063	0.7845	0.7815 ± 0.0127
	LoCoMo (F1)	0.993	0.4074	0.4158	0.4105 ± 0.0020	0.4153	0.4133 ± 0.0033	0.4161	0.4101 ± 0.0038
				$W_{\max}=64$		$W_{\max}=128$		$W_{\max}=512$	
Kimi-K3	AIME 2026	0.8053	0.9656	0.9688	0.9247 ± 0.0083	0.9538	0.9312 ± 0.0145	0.9570	0.9206 ± 0.0069
	HMMT 2026 Feb	0.7235	0.9257	0.9257	0.8970 ± 0.0092	0.9218	0.8964 ± 0.0115	0.9120	0.8952 ± 0.0170
	IMOAnswerBench	0.4616	0.6417	0.6442	0.5977 ± 0.0073	0.6375	0.6025 ± 0.0170	0.6150	0.5667 ± 0.0043
	MMLU-Pro	0.8616	0.8809	0.8806	0.8734 ± 0.0037	0.8793	0.8651 ± 0.0158	0.8753	0.8620 ± 0.0092
	LoCoMo (F1)	0.9988	0.6672	0.6714	0.6511 ± 0.0185	0.6580	0.6479 ± 0.0062	0.6594	0.6230 ± 0.0128

formly retains the same number of units as DASC, holding compression fixed to isolate the benefit of decay-aware ranking. At $W_{\max} = 16$, DASC remains within 1.3 percentage points of dense on every Kimi-Linear and Qwen3-Next row. On Kimi-Linear, it exceeds the five-mask Random mean in all 15 reasoning comparisons. Qwen3-Next remains within 1.7 points of dense, with mixed DASC–Random differences. On Kimi-K3, scores across five tasks stay within 1.2 points of dense at $W_{\max} = 64/128$; at 512, the largest gap is 2.67 points on IMOAnswerBench. DASC exceeds the five-mask Random mean in all 15 point-estimate comparisons (Table 2).

Composition with state quantization. Prior work shows that recurrent states can be quantized alongside model weights and activations (Tianqi et al., 2025). Quantization reduces the bits per retained value, whereas DASC reduces the number of retained units, making the two complementary. We combine $W_{\max} = 16$ channel selection, INT8 state checkpoint storage, and DASC-NR on the five Kimi-Linear end tasks, yielding $8.11\times$ state checkpoint capacity relative to the dense mixed-precision reference. Appendix E.2 gives the protocol.

Table 3: Kimi-Linear DASC at $W_{\max} = 16$ (INT8: $8.11\times$).

Task	Dense	DASC Native	DASC INT8
AIME	0.6065	0.6032	0.6215
HMMT	0.3754	0.3861	0.3715
IMO	0.2783	0.2675	0.2775
GPQA	0.6185	0.6131	0.6128
MMLU	0.6750	0.6732	0.6748

5.3 WildChat serving under matched HBM

We evaluate WildChat-1M serving under matched allocated state-memory budgets (Appendix C.3). Main serving runs use client concurrency 256 and a server concurrency limit of 32. Figure 4 reports five-seed all-request mean TTFT and E2E latency. "TP" enables state checkpoint placement balancing at a fixed TP degree; "Off" disables it.

State checkpoint capacity. Figure 4(a,e,i) shows that capacity depends on model, window, and placement. With TP-balanced placement, Kimi-Linear reaches $2.64/7.06/14.72/28.24\times$ capacity at $W_{\max} = 16/64/128/512$; Qwen3-Next reaches $1.10/1.32/1.60/2.49\times$; and Kimi-K3 reaches $1.61/2.01/3.48\times$ at $W_{\max} = 64/128/512$. The gap reflects selection granularity and layer imbalance: channel-wise KDA exposes more removable units than head-wise GDN.

Serving performance. Figure 4(b,c,f,g,j,k) reports five-seed all-request means, including cache misses and runtime overhead. Selected DASC-NR configurations reduce both TTFT and E2E latency on all three models, while suffix-refresh overhead can offset the capacity benefit. Section 5.4 summarizes the selected configurations and their quality–latency trade-offs.

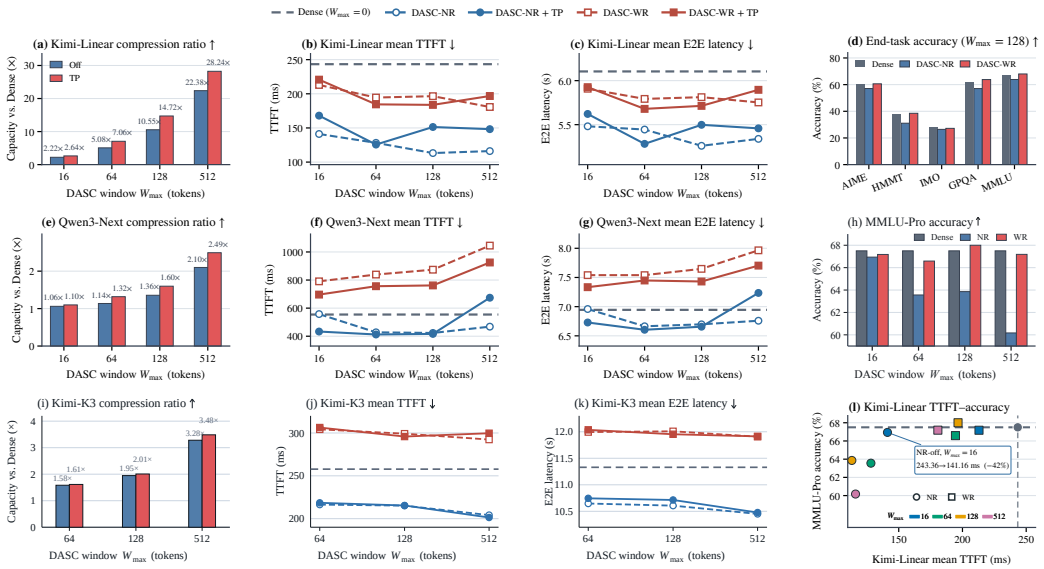


Figure 4: WildChat-1M serving and accuracy; serving panels report five-seed means. Panels (a–c/e–g/i–k) show state checkpoint capacity, mean TTFT, and mean E2E latency for Kimi-Linear/Qwen3-Next/Kimi-K3 (filled: TP-balanced). Panels (d,h) show Kimi-Linear end-task accuracy; (l) pairs its MMLU-Pro accuracy with TTFT. Arrows indicate preferred directions.

5.4 DASC-WR: ACCURACY–LATENCY TRADE-OFF

Figure 4(d,h,l) reports Kimi-Linear accuracy and its trade-off with latency. Panel (l) pairs Kimi-Linear MMLU-Pro accuracy with WildChat TTFT from separate workloads, providing a cross-workload operating-point comparison.

Accuracy recovery. At $W_{\max} = 128$, panel (d) shows that DASC-WR improves all five Kimi-Linear end-task estimates over DASC-NR by 0.8–7.5 percentage points. It matches or exceeds Dense on HMMT, GPQA, and MMLU-Pro and remains within 0.7 points on AIME and IMO. Panel (h) shows the Kimi-Linear MMLU-Pro sweep: DASC-NR ranges from 60.17 to 66.94%, whereas DASC-WR achieves 66.59–68.00%, close to the 67.50% Dense baseline.

Accuracy–latency trade-off. We select close-to-Dense DASC-NR operating points: $W_{\max} = 16$ with Off for Kimi-Linear, $W_{\max} = 64$ with TP balancing for Qwen3-Next, and $W_{\max} = 64$ with Off for Kimi-K3. Relative to Dense, these reduce mean TTFT by $42.0/25.6/16.1\%$ and mean E2E latency by $10.2/4.9/6.0\%$, respectively. The corresponding quality differences are 0.56 points on Kimi-Linear MMLU-Pro, at most 0.92 points across six Qwen3-Next end tasks, and at most 0.42 points across five Kimi-K3 end tasks. These cross-workload operating points balance quality and latency: larger windows risk quality loss, while WR adds recovery cost. Keeping the selected Kimi-K3 configuration fixed, we additionally test server concurrency limits of 16 and 8 at client concurrency 256. Under increased queueing pressure, mean TTFT falls by $15.7/11.3\%$ and mean E2E latency by $11.3/9.2\%$, respectively (Appendix Table 4).

6 CONCLUSION

We presented DASC, combining decay-aware selection, ragged storage, and optional TP-balanced placement for hybrid prefix caches. Conservative configurations retain quality close to Dense. At the selected operating points in Section 5.4, mean TTFT falls by $42.0/25.6/16.1\%$ and mean E2E latency by $10.2/4.9/6.0\%$ on Kimi-Linear/Qwen3-Next/Kimi-K3, respectively, demonstrating serving benefits across channel-wise KDA and head-wise GDN.

AI USE STATEMENT

Generative AI tools were used to improve manuscript wording and readability and to assist with figure formatting and visual refinement. The authors reviewed all AI-assisted edits and verified that they preserved the intended technical meaning and reported data. The authors take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies recurrent-state checkpoint compression using existing models and datasets, without recruiting human participants or collecting new conversations. Use of the models and conversation datasets remains subject to their licenses and privacy restrictions. Compression can alter model outputs, and the reported efficiency gains do not establish improved safety or reliability. Deployments should account for the underlying models' limitations and evaluate the effects of compression in their intended applications.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the static-horizon construction, state selection, storage layouts, and load policies. Section 5.1 identifies the evaluated models, benchmarks, and baselines. Appendix A.2 details horizon validation, and Appendix B describes TP balancing and two-tier cache management. Appendix C documents task-specific generation settings, warmup–replay evaluation, seed aggregation, and matched-memory serving protocols. Appendices E.1 and E.2 describe state quantization and its composition with DASC.

REFERENCES

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms, 2026. URL https://arxiv.org/abs/2605.00674.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient large language model serving for multi-turn conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 111–126, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-41-0. URL https://www.usenix.org/conference/atc24/presentation/gao-bin-cost.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303. Curran Associates, Inc., 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *ACM Transactions on Computer Systems*, 44(1):1–27, November 2025. ISSN 1557-7333. doi: 10.1145/3768628. URL http://dx.doi.org/10.1145/3768628.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming

Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626. ACM, October 2023. doi: 10.1145/3600006.3613165. URL http://dx.doi.org/10.1145/3600006.3613165.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

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, 2024. URL https://arxiv.org/abs/2403.19887.

Ze-Wei Liou and Tri Dao. ReplaySSM: Cache SSM inputs, not state. Dao AI Lab Blog, June 2026. URL https://dao-lab.ai/blog/2026/replayssm/.

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. Towards robust mathematical reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35418–35442, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

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 M. Zaharia, G. Joshi, and Y. Lin (eds.), *Proceedings of Machine Learning and Systems*, volume 7. MLSys, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/7c180af017258d239bac6248d1eb26ac-Paper-Conference.pdf.

Qwen Team. Qwen3-Next: Towards ultimate training & inference efficiency. Qwen Blog, 2025. URL https://qwen.ai/blog?id=4074cca80393150c248e508aa62983f9cb7d27cd.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

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.

Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 53551–53575, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/84a7fc24ed52e8eff514c33e8ac76ea3-Paper-Conference.pdf.

Chen Tianqi, Yuanteng Chen, Peisong Wang, Weixiang Xu, Zeyu Zhu, and Jian Cheng. Q-mamba: Towards more efficient mamba models via post-training quantization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10594–10610, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.551. URL https://aclanthology.org/2025.findings-acl.551/.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522. Curran Associates, Inc., 2024. doi: 10.52202/079017-3668. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d13a3eae72366e61dfdc7eea82eeb685-Paper-Conference.pdf.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4904fad153f6434a7bcf04465d4be2cc-Paper-Conference.pdf.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 34590–34605, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9421261e06f1a63a352b068f1ac90609-Paper-Conference.pdf.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

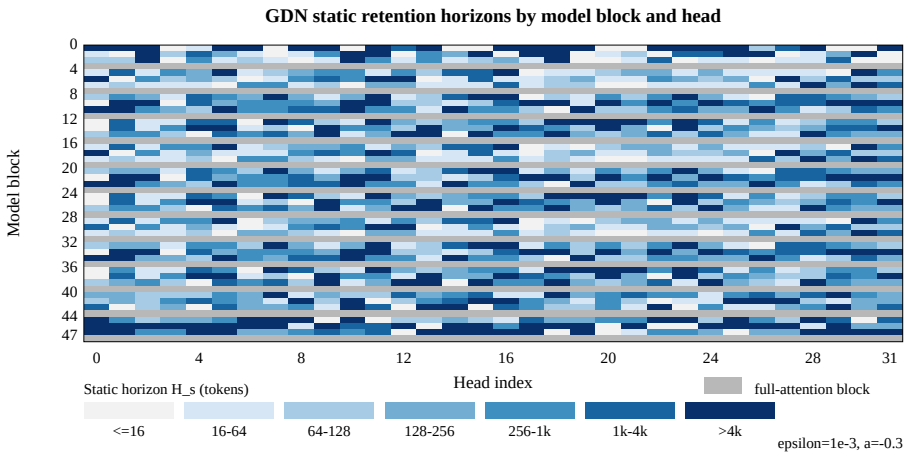


Figure 5: Static horizons of all Qwen3-Next GDN heads ($a = -0.3$, $\epsilon = 10^{-3}$), in model-block/head order. Gray rows are full-attention blocks.

A Additional retention-horizon analysis

A.1 Layerwise structure

Figures 5–8 preserve architectural ordering, complementing the pooled decay profiles in Figure 1. Qwen3-Next's GDN has one horizon per head; Kimi-Linear and Kimi-K3 additionally exhibit within-head channel variation. Horizon bins summarize ranges rather than every tested cutoff.

For either KDA model, the retained-channel fraction of layer l and head h at cutoff w is

$$r_{l,h}(w) = \frac{1}{128} \sum_{k=1}^{128} \mathbf{1}[H_s(l,h,k) > w] \,.$$

A layer–head pair is mixed when $0 < r_{l,h}(w) < 1$. The mixed-pair percentage is pooled across all KDA layers and heads, not measured in a single representative layer. Each layer's mean retained fraction averages $r_{l,h}$ over its heads; the ranges in the captions are the minimum and maximum of these layer means. Gray full-attention rows are excluded throughout. Kimi-Linear uses 20 KDA layers with 32 heads each, $a_s = -0.3$, and $w = 16$; Kimi-K3 uses 69 KDA layers with 96 heads each, $a_s = 0.8$, and $w = 64$. Both use $\epsilon = 10^{-3}$ and their native gates from Eq. 2.

These weight-derived maps are independent of serving requests. Within-head variation motivates channel-wise selection, while unequal layerwise retained counts motivate ragged storage. The maps do not measure state/readout correlation or establish that omitted channels are unimportant; downstream quality experiments evaluate the effect of omission.

A.2 Static-horizon validation

For Qwen3-Next and Kimi-Linear (Table 1(b)), we deterministically select 32 prefixes per model from the unfiltered cleaned ShareGPT V3 release (`anon8231489123/ShareGPT_Vicuna_unfiltered`; 94,145 conversations). Turns are serialized as alternating `User:` and `Assistant:` blocks. We process target lengths 16,384, 8,192, 4,096, and 1,024 in descending order so that long conversations are not consumed by shorter tiers, selecting eight prefixes per tier. For each target T, we first retain conversations with at least $3.5T$ characters, tokenize the first 500 eligible candidates, sort them by token count, and deterministically choose the eight shortest candidates with at least T tokens. If fewer than eight qualify, we use the longest available candidates, possibly shorter than T. Longer sequences are left-truncated to T tokens. Each model independently selects samples with its own tokenizer, so raw conversation sets may differ; source-array indices are released with the analysis output. No random

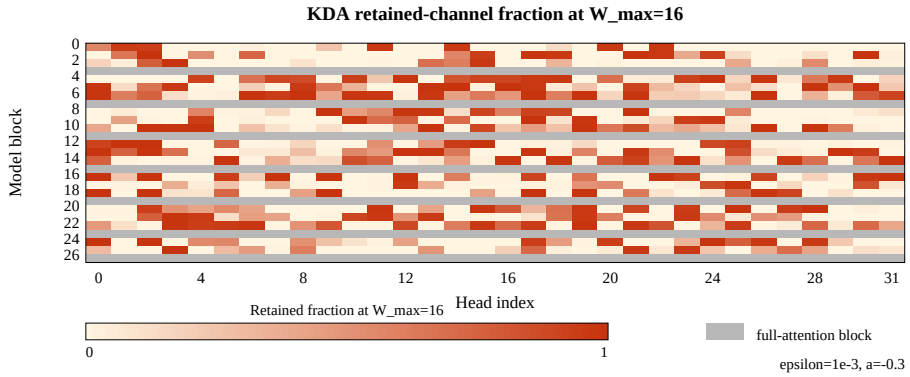


Figure 6: Kimi-Linear retained-channel fractions at a 16-token cutoff. Each cell summarizes 128 channels. Across 20 KDA layers and 32 heads, 389 of 640 layer–head pairs (60.8%) contain both retained and omitted channels, 215 omit all channels, and 36 retain all. Layerwise mean retained fractions range from 0.179 to 0.558; gray full-attention rows are excluded.

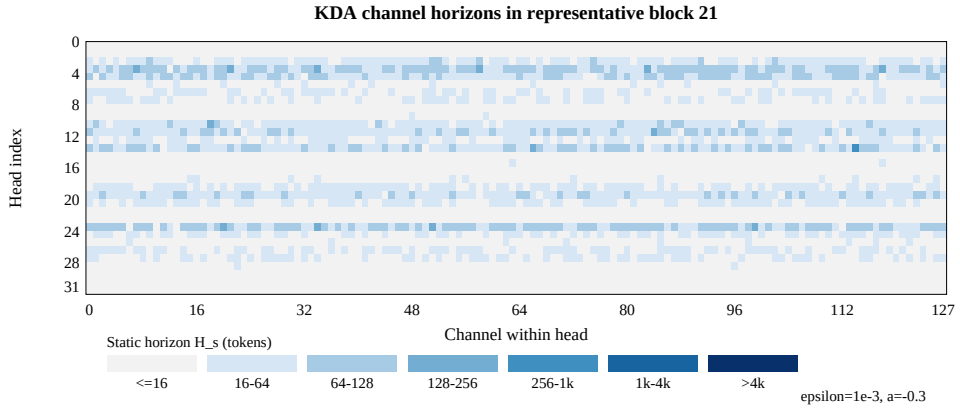


Figure 7: Channel horizons in Kimi-Linear block 21, whose retained fraction (0.375) is closest to the layer median (0.360). Indices retain architectural order.

sampling seed is used. Each prefix is processed independently; prefixes are not concatenated. We run a full BF16 forward pass through Qwen3-Next (36 GDN layers) and Kimi-Linear (20 KDA layers), hook the observed token-dependent gates in every linear-attention layer, and execute the corresponding delta-rule recurrence to obtain the recurrent state magnitude at the final prefix position. No analysis prefix is used to construct the deployed mask. Kimi-K3 uses 32 ShareGPT prefixes spanning 128–2,048 tokens, processed independently through all 69 KDA layers (96 heads $\times$ 128 key channels), for 2,208 prefix–layer observations across the full set. The even-indexed 16 prefixes select $a_s = 0.8$, and the odd-indexed 16 provide held-out validation; this calibration split and shorter prefix range differ from the other two models. These shorter-prefix measurements characterize gate ordering and final-state activity on that range; they do not establish long-context calibration validity. Longer-context quality is assessed separately, rather than inferred from these correlations.

For all three models, empirical horizons and state magnitudes follow Eqs. 4 and 5. Static horizons use $\epsilon = 10^{-3}$ and each model's native gate in Eq. 2, with $a_s = -0.3$ for Qwen3-Next and Kimi-Linear and $a_s = 0.8$ for Kimi-K3. We calculate Spearman correlations across units separately for each validation prefix and layer, then report their median and IQR in Table 1(b). Units from different layers are not pooled.

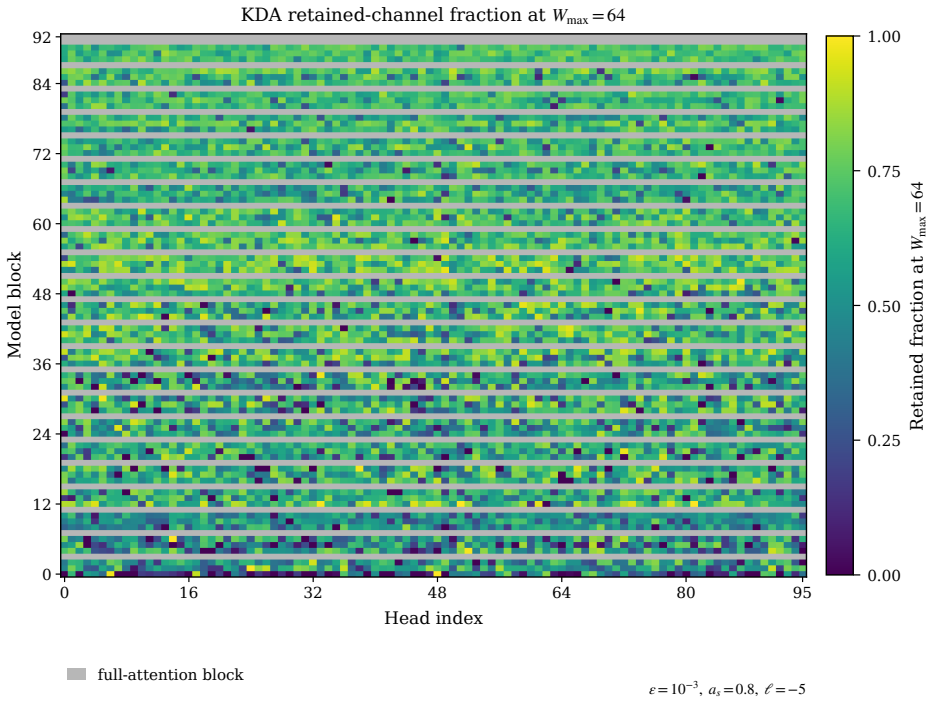


Figure 8: Kimi-K3 retained-channel fractions at a 64-token cutoff. Across 93 model blocks, 69 KDA layers contribute 6,624 layer–head pairs; gray full-attention rows are excluded. Of these pairs, 6,512 (98.3%) contain both retained and omitted channels, 108 omit all channels, and 4 retain all. Layerwise mean retained fractions range from 0.267 to 0.741.

B Buffer layout details

Padded (Scheme 1). Let $R_{\max} = \max_{1\le l\le L} r_l$, where L is the number of recurrent linear-attention layers. For GDN, where one retained unit is a complete head state, the pool is $[L, \text{slots}, R_{\max}, d_v, d_k]$. For KDA, where one retained unit is a (head, k) column, it is $[L, \text{slots}, R_{\max}, d_v]$. Only the first r_l entries of layer l are valid; the remaining entries are padding. A single wide layer thus pins every layer's temporal allocation, giving $C^{\text{pad}}_{\text{temp}} = (\sum_{l=1}^{L} n_l)/(LR_{\max})$. Full state checkpoint compression additionally includes the BF16 convolution state as in Eq. 6.

Ragged (Scheme 2). Each layer contributes only its own r_l retained units, with offsets $o_0 = 0$ and $o_l = o_{l-1} + r_l$ for $l = 1, \ldots, L$. GDN uses $[\text{slots}, \sum_{l=1}^{L} r_l, d_v, d_k]$, whereas KDA uses $[\text{slots}, \sum_{l=1}^{L} r_l, d_v]$; layer l occupies $[o_{l-1}, o_l)$. With $\overline{r} = \frac{1}{L}\sum_{l=1}^{L} r_l$, the temporal layout achieves $C^{\text{rag}}_{\text{temp}} = (\sum_{l=1}^{L} n_l)/(L\overline{r})$. The retained values match the padded layout, so accuracy is layout-independent. The resulting temporal-layout gain, $C^{\text{rag}}_{\text{temp}}/C^{\text{pad}}_{\text{temp}} = R_{\max}/\overline{r}$, increases with layer imbalance: unequal retained-unit counts create padding overhead that the ragged layout avoids.

B.1 Deterministic TP balancing

The compute layout and the persistent state checkpoint layout are distinct. Figure 9 specifies both offline assignment procedures; the resulting maps are reused at runtime. Let P be the TP degree. The cost $c_{l,h}$ of head h in layer l is its retained-channel count for KDA and its retained-head indicator for GDN. Within each store, equal-size units make unit-count and byte-based ordering equivalent. A bundle comprises all coupled tensors of a KDA head ($B = 1$), or the $B = H_v/H_k$ adjacent GDN value heads sharing one key head. Its cost is $C_{l,b} = \sum_{h=bB}^{(b+1)B-1} c_{l,h}$. The implementation requires $H_v \bmod B = 0$ and $(H_v/B) \bmod P = 0$, assigning exactly $K = (H_v/B)/P$ bundles per rank per layer. Cumulative rank loads L_r include all processed layers.

(a) Compute-head permutation

Input: bundle costs $C_{l,b}$, TP degree P, per-layer quota K.
Output: bundle owners and loads L_r.

1. Initialize $L_r = 0$ for all ranks.
2. For each layer l in increasing order:

   Set per-rank bundle counts $u_r = 0$.

   Sort bundles by $(-C_{l,b}, b)$.

   For each bundle b, among $u_r < K$, select the lexicographically smallest

   $$(L_r + C_{l,b},\ L_r,\ u_r,\ r).$$

   Assign b to that rank; increment u_r and add $C_{l,b}$ to L_r.
3. Define the load objective

   $$\Phi(\mathbf{L}) = \big(\max_r L_r,\ \sum_r L_r^2\big).$$

4. Scan layers and pairs $b_1 < b_2$ in fixed increasing order. Only consider pairs on different ranks in the same layer.
5. Evaluate Φ after each swap; retain the best strict improvement. Keep the first candidate on equal scores.
6. If no improving swap exists, stop. Otherwise apply it and repeat step 4.
7. Apply the resulting permutation consistently to every head-coupled tensor, including output-projection input columns.

Guarantee: per-layer quotas are preserved; the final assignment is locally optimal for a single within-layer pair swap, not globally optimal.

(b) State checkpoint storage balancing

Input: compute-local unit counts n_r after optional permutation.
Output: storage-owner map and transfer splits.

1. Compute $N = \sum_r n_r$ and $(q, m) = \mathrm{divmod}(N, P)$.
2. Sort ranks by $(-n_r, r)$; set $t_r = q + 1$ for the first m ranks and $t_r = q$ for the rest.
3. Keep the first $\min(n_r, t_r)$ local units on each rank.
4. List excess units by source rank, then local unit index. List deficit positions by destination rank, then position.
5. Pair both lists in order; precompute the destination map and variable split sizes.
6. Allocate $\lceil N/P \rceil$ units per slot on every rank.

Runtime store
Compute-local state → fused pack/partition → variable-split all-to-all → scatter into storage slots.

Runtime load
Storage slots → fused gather/partition → inverse all-to-all → scatter into compute-local active state.

Transfer volume:

$$M_{\text{remote}} = \sum_r \max(0, n_r - t_r).$$

For these targets, each excess unit must move and moves once; transfer is minimal for equal-size units. If $M_{\text{remote}} = 0$, skip the all-to-all.

Figure 9: Deterministic TP assignment algorithms. (a) Optional compute-side permutation preserves equal bundle counts. (b) Storage balancing changes storage ownership independently of compute ownership. All comparisons and tie-breaking are deterministic.

Initialization versus runtime communication. The compute permutation moves Q/K/V and gate projections, recurrent decay parameters, convolution parameters, output-projection input columns, and other coupled axes together. Initialization may use collectives, but subsequent head-separable recurrence uses the locally owned weights and active states. Existing TP collectives remain; permutation adds no token-time collective. Storage balancing leaves compute ownership unchanged, so remotely stored units still require communication at each state checkpoint store/load boundary. Fused/batched operations reduce launches and temporary buffers, not the logically required remote payload. In implementation, `off` changes neither compute nor cross-rank storage ownership; `storage` applies storage balancing only; and `hybrid` adds compute-head permutation before storage balancing. The compute permutation requires unquantized linear weights, so Kimi-K3's MXFP4 packed weights use `storage` only. This compatibility constraint concerns model weights, not state checkpoint precision. The experimental label *TP* denotes balanced placement collectively, rather than a particular implementation mode.

B.2 Two-tier state checkpoint management

Allocation and ownership. The two-tier integration uses a native Mamba slot allocator for both running states and cached dense state checkpoints, plus a separately allocated, fixed-capacity DASC

compressed pool. Dense cached slots can therefore be reclaimed for active computation. Radix nodes record the representation tier, slot identifier, and lock reference count; dense and compressed cached nodes share one global Mamba LRU order. `UnifiedRadixCache` unifies cache metadata and lifecycle management, not raw HBM allocation: this integration does not place KV, dense Mamba state, and compressed DASC state in one `UnifiedKVPool`. The unified-HBM construction path still sets `mamba_ckpt_pool=None`.

Insertion and reuse. DASC precomputes selection masks, ragged offsets, and optional TP placement maps once and reuses them for state checkpoint packing and restoration. Runtime insertion is compressed-first. When a compressed slot is available, the runtime packs the selected units and releases the dense source only after the transfer completes and ownership permits release. When the compressed pool is full, an eligible completed request transfers its dense active-slot ownership to Radix without an additional copy. A dense hit uses an exact batched copy-on-write into active state and bypasses DASC recovery. A compressed hit restores retained global units; DASC-NR zero-fills omitted units, whereas DASC-WR refreshes them using the bounded scratch re-prefill in Section 4.2. The remaining request suffix then executes normally. Placement collectives occur only at compressed state checkpoint store/load boundaries; the load transfer inverts the store mapping, and token-wise recurrence remains rank-local.

Pressure-driven reclamation. When free native slots fall below the reclamation watermark, the runtime considers only cached nodes that are unlocked and unreferenced by running requests. If compressed space is available, it demotes the oldest eligible dense state checkpoint. If both pools are full and the oldest eligible global-LRU node is dense, it evicts that node directly. If that node is compressed, the runtime evicts it and uses the freed compressed slot to demote the oldest eligible dense node. Demotion preserves the node's original LRU position: changing representation is not a cache access. Evicting a compressed node alone cannot free a native slot. With no eligible dense source, this reclamation path cannot satisfy active-state demand; scheduling must wait or apply backpressure. KV eviction is useful only where the allocator actually permits the released memory to satisfy that allocation.

Transactional demotion. Demotion pins the Mamba component, packs on a background CUDA stream, and records a completion event. Until completion, the dense source remains valid and cannot be reused. The runtime commits the new tier and slot and releases the source only after the event completes and concurrent cache-hit locks clear. Logically, a node passes through dense, demoting, compressed, and absent states. Victim selection and slot mappings must agree across TP ranks, and batched reclamation must preserve this ordering. A dense-victim index may accelerate selection, but it must use the same recency ordering rather than define a second eviction policy.

Memory boundary and costs. The fixed compressed allocation cannot be borrowed directly for KV or running states. Dense spill retains elasticity within the native Mamba allocation, while watermark demotion can reduce request-path stalls without eliminating pack, transfer, and event costs. A total-HBM comparison must charge native-state capacity, compressed storage, staging/scratch buffers, and KV allocations to the same budget in both systems; compressed capacity must not be added on top of an unchanged native budget. The WildChat serving experiments use this integration; Appendix C distinguishes allocated pool capacity from runtime residency.

C Evaluation protocol

C.1 Shared system and state checkpoint accounting

All formal experiments run in SGLang. Kimi-Linear and Qwen3-Next use eight Hopper-architecture GPUs with tensor parallelism (TP) 8; their model execution and weights use BF16 without model quantization. Recurrent state checkpoints use SGLang's default mixed precision: FP32 temporal state and BF16 convolution state. The evaluated configuration uses `mem-fraction-static`=0.80, the overlap scheduler disabled, and cache telemetry enabled. WildChat uses the two-tier state checkpoint integration in Appendix B.2: native active states and dense spill share a Mamba slot allocator, while a separate fixed pool stores ragged compressed state

checkpoints. Both cached representations share one global Mamba LRU. Dense retains full mixed-precision state checkpoints without DASC packing. Across arms, model weights, tokenizer and chat template, frozen input IDs, prompt hashes, request order, and sampling seeds are identical.

Matched-HBM serving charges both resident tiers and temporary state buffers to the allocated state-memory budget. With unchanged KV and model allocations, the per-rank comparison satisfies

$$B^{\text{Dense}}_{\text{native}} + B^{\text{Dense}}_{\text{tmp}} = B^{\text{DASC}}_{\text{native}} + B^{\text{DASC}}_{\text{compressed}} + B^{\text{DASC}}_{\text{tmp}}.$$

Here native memory includes running states and dense cached state checkpoints; temporary memory includes staging and scratch allocations. The fixed compressed pool is not additional memory on top of an unchanged Dense native allocation. Physical state checkpoint size includes FP32 temporal state and uncompressed BF16 convolution state, and TP capacity is limited by the bottleneck rank.

For a separately allocated pool of per-rank budget B_{pool} and physical bytes per slot b^{full}_r, its usable slot count is $\lfloor B_{\text{pool}} / \max_r b^{\text{full}}_r \rfloor - 1$, reserving allocator slot 0. This is an allocation-capacity calculation, not a count of all resident prefixes: dense spill can occupy otherwise free native slots, subject to active-request demand. Compression ratios describe the state checkpoint representation or configured capacity, not an equal reduction in total process HBM. Full-attention KV remains unchanged and is excluded from these compression ratios.

C.2 Quality-evaluation protocols

Compressed-only state checkpoint caching. In DASC and Random accuracy runs, all cached recurrent state checkpoints reside in the compressed pool; dense checkpoint spill is disabled. Native dense slots are used for active computation, not checkpoint caching, so a state checkpoint hit cannot bypass compression through a dense cached copy. The Dense baseline retains full state checkpoints. This restriction does not imply that every replay request is a cache hit.

Shared warmup–replay accounting. RULER and Table 2 first issue an unscored warm request for each evaluated prefix and then issue measured replay requests. Warmup outputs and tokens are discarded. Accuracy or F1 includes every measured replay output rather than conditioning on a successful cache hit; Table 2 reports Hit pooled across all compared arms in each row, whose hit rates are similar, $H = \sum_i C_i / \sum_i (C_i + N_i)$, where i ranges over measured replay requests across these arms and C_i/N_i count cached/new input tokens. Replays are averaged within each unique instance or problem, which remains the statistical unit, and paired arms share prompts, warmup/replay order, seeds, and slot counts.

RULER. Each subtask–length–arm setting uses $N = 30$ unique instances and three measured replay rounds. The complete per-subtask sweeps are in Tables 5 and 6.

Controlled suffix-refresh accuracy. Figure 4(d,h) follows the shared warmup–replay accounting with additional cache-residency controls. Panel (d) reports Kimi-Linear replay accuracy for five reasoning benchmarks at $W_{\max} = 128$; panel (h) reports the Kimi-Linear MMLU-Pro sweep over $W_{\max} \in \{16, 64, 128, 512\}$. Panel (l) pairs Kimi-Linear MMLU-Pro replay accuracy with WildChat TTFT from a separate workload, so it is a cross-workload operating-point view rather than a single-run metric.

The MMLU-Pro protocol fixes 512 state checkpoint slots, 192 questions per block, concurrency 256, and prompt-only state checkpointing, so Dense, DASC-NR, and DASC-WR retain the same prompt boundaries and continuation states cannot evict them. Each block is flushed, filled by one warm request per question (`try_id`=0), checked for at least 272 available slots, and replayed seven times per question (`try_id`=1,...,7). Each block must have zero replay eviction, token hit > 0.70, and at least 95% loaded requests; each complete arm must have at least 99% aggregate loaded requests.

Accuracy includes all measured replay outputs, regardless of cache-hit status, and excludes warmup outputs. Paired deltas are computed on the same full set of measured (question, try) pairs across arms, without filtering to successfully loaded pairs. Kimi-Linear uses temperature 1.0 and top-p 1.0 and evaluates all 12,032 MMLU-Pro questions with one warm and seven measured replays per question.

Five-benchmark end-task suite. Table 2 uses the shared two-stage schedule but not the additional block-residency qualification above; its measured replay requests run through the standard concurrent serving path. We evaluate all unique problems in AIME 2026 ($N = 30$), HMMT 2026 Feb ($N = 33$), IMOAnswerBench ($N = 400$), GPQA-Diamond ($N = 198$), and MMLU-Pro ($N = 12{,}032$). Each problem receives one unscored warmup followed by 31, 31, 3, 15, or 7 measured replays for AIME, HMMT, IMOAnswerBench, GPQA-Diamond, or MMLU-Pro, respectively (32, 32, 4, 16, or 8 total generations). Only measured replay outputs contribute to accuracy. Kimi-K3 evaluates AIME, HMMT, IMOAnswerBench, and MMLU-Pro with the same semantics, and LoCoMo under the protocol below. Its DASC entries are point estimates and its Random entries are five-mask mean$\pm$sample SD. All three models use the same count-matched Random mask seeds $\{1234, 2027, 3407, 4519, 5683\}$.

LoCoMo. Each arm evaluates the same $1{,}986$ questions using one measured greedy replay per question after unscored cache population of the shared serialized conversation prefix. We uniformly apply ordinary token F1, without category-specific scoring, and average across questions; each question is the statistical unit.

C.3 WildChat matched-HBM performance protocol

Serving performance is evaluated using natural conversations from WildChat-1M (Zhao et al., 2024). Each model uses a deterministic, tokenizer-specific trace containing 256 four-turn conversations, or 1,024 requests. Initial prompts contain 1k–8k tokens, final contexts are capped at 16k tokens, and decode length follows the recorded WildChat assistant-response length, clipped to 1,024 tokens. No synthetic prefix concatenation is used. Unlike the compressed-only accuracy runs, DASC serving enables compressed-first insertion, dense spill, and global-LRU demotion as described in Appendix B.2.

The matrix compares Dense with DASC-NR/DASC-WR, with and without TP-balanced placement, at $W_{\max} \in \{16, 64, 128, 512\}$ for Kimi-Linear and Qwen3-Next and $W_{\max} \in \{64, 128, 512\}$ for Kimi-K3. Within a model, all arms use the same allocated state-memory budget, including native slots, compressed storage, and temporary buffers. Full-attention KV allocation is unchanged and excluded from the state checkpoint compression ratio. All three models use the same five run seeds. For each seed, TTFT and E2E latency are averaged over all 1,024 requests rather than conditioned on cache hits; Figure 4 reports the mean across seeds. The measurements therefore include misses, state checkpoint lookup and load, prefill, decoding, and DASC runtime overhead.

Kimi-K3 under increased queueing pressure. The main serving experiments use client concurrency 256 and a server concurrency limit of 32, with no appreciable queueing. We additionally evaluate the selected Kimi-K3 DASC-NR configuration at $W_{\max} = 64$ with Off, keeping client concurrency at 256 and lowering the server concurrency limit to 16 or 8. These settings increase queueing pressure rather than the configured client concurrency. Each setting is evaluated over five seeds. Table 4 reports mean and P95 TTFT and E2E latency. Each statistic is computed separately within each seed and then averaged across the five seeds; P95 is not computed from pooled requests across seeds. At server limits 16/8, mean TTFT decreases by $15.70/11.26\%$ and mean E2E latency by $11.25/9.22\%$, respectively. Both P95 metrics also improve, supporting the selected configuration under the tested queueing conditions without implying optimality at every load.

Table 4: Kimi-K3 latency under increased queueing pressure (seconds; lower is better). Client concurrency is 256; each setting uses five seeds. DASC uses DASC-NR at $W_{\max} = 64$ with Off.

Server limit	Method	TTFT (s)		E2E latency (s)	
		Mean	P95	Mean	P95
16	Dense	18.392	50.946	27.238	62.450
	DASC-NR	15.505	41.964	24.174	53.157
8	Dense	54.453	93.929	76.547	131.924
	DASC-NR	48.323	88.864	69.488	126.528

D COMPLETE QUALITY RESULTS

Tables 5 and 6 report the complete Kimi-Linear and Qwen3-Next RULER sweeps, respectively.

Table 5: Full replay-only Kimi-Linear (KDA) RULER sweep ($N = 30$ per subtask–length; replay only).

	Arm	Len	S1	S2	S3	MK1	MK2	MK3	MQ	MV	CWE	FWE	QA-H	QA-S	VT	**Avg**
baseline	dense	4k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.89	0.57	0.90	1.00	0.95
		8k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.62	0.86	1.00	0.96
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.94	0.57	0.93	0.98	0.95
DASC-NR	$W_{\max}16$	4k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.89	0.59	0.90	1.00	0.95
		8k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.59	0.87	1.00	0.95
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.56	0.88	0.99	0.95
	$W_{\max}64$	4k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00	0.89	0.58	0.92	1.00	0.95
		8k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.99	0.93	0.59	0.78	1.00	0.94
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	0.95	0.54	0.84	0.99	0.95
	$W_{\max}128$	4k	1.00	1.00	0.98	1.00	1.00	1.00	1.00	0.97	0.89	0.89	0.53	0.91	1.00	0.94
		8k	1.00	1.00	0.97	1.00	1.00	1.00	1.00	0.95	0.91	0.93	0.57	0.84	1.00	0.94
		16k	1.00	1.00	0.97	0.98	1.00	1.00	1.00	0.97	0.82	0.94	0.53	0.92	1.00	0.93
	$W_{\max}512$	4k	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.89	0.85	0.86	0.47	0.87	0.99	0.92
		8k	1.00	1.00	1.00	0.96	1.00	0.97	1.00	0.86	0.70	0.91	0.62	0.84	0.98	0.91
		16k	1.00	1.00	1.00	0.88	0.97	0.98	1.00	0.89	0.66	0.94	0.57	0.84	0.96	0.90
	$W_{\max}1024$	4k	1.00	0.99	1.00	1.00	1.00	1.00	0.98	0.87	0.75	0.86	0.53	0.86	1.00	0.91
		8k	1.00	1.00	1.00	0.96	0.97	0.94	1.00	0.85	0.76	0.92	0.59	0.80	0.96	0.90
		16k	1.00	0.98	1.00	0.82	0.94	0.98	1.00	0.83	0.72	0.94	0.52	0.79	0.92	0.88
DASC-WR	$W_{\max}16$	4k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.89	0.59	0.90	1.00	0.95
		8k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.57	0.86	1.00	0.95
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.94	0.57	0.87	0.99	0.95
	$W_{\max}64$	4k	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.96	1.00	0.89	0.57	0.90	1.00	0.95
		8k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95	1.00	0.93	0.60	0.82	0.99	0.95
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.91	0.94	0.58	0.87	0.96	0.94
	$W_{\max}128$	4k	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.91	0.95	0.88	0.56	0.90	1.00	0.94
		8k	1.00	1.00	1.00	0.97	1.00	1.00	1.00	0.88	0.95	0.93	0.60	0.84	0.98	0.93
		16k	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.85	0.87	0.94	0.58	0.87	0.99	0.93
	$W_{\max}512$	4k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.89	0.96	0.88	0.57	0.90	1.00	0.94
		8k	1.00	1.00	1.00	0.94	0.99	1.00	1.00	0.82	0.98	0.93	0.59	0.84	0.99	0.93
		16k	1.00	1.00	1.00	0.96	0.98	1.00	1.00	0.84	0.81	0.94	0.64	0.88	0.99	0.93
	$W_{\max}1024$	4k	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.96	0.97	0.88	0.57	0.90	1.00	0.94
		8k	1.00	1.00	1.00	0.94	0.99	1.00	1.00	0.86	0.97	0.93	0.60	0.82	1.00	0.93
		16k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.96	0.94	0.64	0.88	1.00	0.95

Table 6: Full replay-only Qwen3-Next (GDN) RULER sweep ($N = 30$ per subtask–length).

	Arm	Len	S1	S2	S3	MK1	MK2	MK3	MQ	MV	CWE	FWE	QA-H	QA-S	VT	**Avg**
baseline	dense	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.97	0.39	0.98	0.57	0.92	0.37	0.84
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.84	0.35	0.97	0.62	0.82	0.40	0.83
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.73	0.29	0.93	0.51	0.83	0.41	0.80
DASC-NR	$W_{\max}16$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.98	0.38	0.97	0.57	0.92	0.38	0.84
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.86	0.34	0.97	0.60	0.84	0.40	0.83
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.74	0.31	0.93	0.53	0.84	0.42	0.81
	$W_{\max}64$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.97	0.36	0.97	0.54	0.92	0.36	0.84
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.85	0.37	0.97	0.61	0.82	0.39	0.83
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.74	0.28	0.93	0.50	0.82	0.42	0.80
	$W_{\max}128$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.94	0.40	0.97	0.57	0.92	0.38	0.84
		8k	1.00	1.00	1.00	1.00	1.00	0.99	0.75	0.85	0.40	0.97	0.61	0.82	0.41	0.83
		16k	1.00	1.00	1.00	1.00	1.00	0.94	0.75	0.73	0.30	0.93	0.53	0.84	0.43	0.80
	$W_{\max}512$	4k	1.00	1.00	1.00	1.00	1.00	0.97	0.75	0.96	0.30	0.97	0.54	0.92	0.36	0.83
		8k	1.00	1.00	1.00	1.00	1.00	0.96	0.75	0.86	0.30	0.97	0.58	0.82	0.39	0.82
		16k	1.00	1.00	1.00	1.00	1.00	0.89	0.75	0.75	0.24	0.93	0.53	0.87	0.40	0.80
	$W_{\max}1024$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.73	0.94	0.35	0.97	0.56	0.92	0.35	0.83
		8k	1.00	1.00	1.00	1.00	1.00	0.97	0.75	0.86	0.35	0.97	0.62	0.84	0.40	0.83
		16k	1.00	1.00	1.00	1.00	1.00	0.87	0.75	0.75	0.27	0.93	0.49	0.84	0.38	0.79
DASC-WR	$W_{\max}16$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.96	0.20	0.97	0.56	0.92	0.37	0.83
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.85	0.54	0.97	0.60	0.84	0.39	0.84
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.75	0.43	0.93	0.51	0.84	0.42	0.82
	$W_{\max}64$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.94	0.23	0.98	0.57	0.92	0.38	0.83
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.86	0.53	0.97	0.61	0.82	0.40	0.84
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.72	0.34	0.93	0.51	0.83	0.42	0.81
	$W_{\max}128$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.95	0.24	0.97	0.54	0.92	0.38	0.83
		8k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.85	0.54	0.97	0.58	0.82	0.40	0.84
		16k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.76	0.37	0.93	0.50	0.84	0.41	0.81
	$W_{\max}512$	4k	1.00	1.00	1.00	1.00	1.00	1.00	0.75	0.95	0.21	0.97	0.54	0.92	0.38	0.82
		8k	1.00	1.00	1.00	1.00	1.00	0.98	0.75	0.86	0.54	0.97	0.60	0.82	0.39	0.84
		16k	1.00	1.00	1.00	1.00	0.98	0.90	0.75	0.74	0.43	0.93	0.53	0.84	0.43	0.81
	$W_{\max}1024$	4k	1.00	1.00	1.00	1.00	1.00	0.99	0.73	0.94	0.24	0.97	0.57	0.92	0.39	0.83
		8k	1.00	1.00	1.00	1.00	1.00	0.97	0.75	0.83	0.43	0.97	0.61	0.82	0.40	0.83
		16k	1.00	1.00	1.00	1.00	0.98	0.83	0.75	0.76	0.35	0.93	0.50	0.83	0.48	0.80

E STATE QUANTIZATION AND DASC COMPOSITION

E.1 INT8 STATE CHECKPOINT PROTOCOL

This state-quantization study evaluates Kimi-Linear only. Its unquantized Dense and DASC arms use SGLang's default mixed-precision recurrent state checkpoint: FP32 temporal state and BF16 convolution state. Model weights and execution remain BF16 and are not quantized. For INT8 arms, we quantize only the cached FP32 temporal recurrent state S; convolution state remains BF16 and the active temporal state used by recurrence remains FP32. Quantization is symmetric signed INT8 over $[-127, 127]$, without a zero point. For a state tensor $S \in \mathbb{R}^{L\times H\times d_v\times d_k}$, one scale is used for each (l, h, k) tuple and shared over d_v:

$$a_{l,h,k} = \max_v |S_{l,h,v,k}|, \qquad s_{l,h,k} = \frac{\max(a_{l,h,k}, 10^{-8})}{127}, \qquad q = \mathrm{clip}(\mathrm{round}(S/s), -127, 127)\,. \tag{8}$$

The maximum, division, and rounding are computed in FP32; q is stored as INT8 and s in the temporal source dtype (FP32 here). On a cache hit, the state checkpoint is dequantized once as $\hat{S} = q\,s$ into a fresh active FP32 temporal-state slot; subsequent recurrence and decoding use native precision. Thus state checkpoints are quantized once at insertion and dequantized once on reuse, rather than repeatedly quantized during recurrence. Conv1D window states remain BF16 because their footprint is small. Reported HBM includes the INT8 payload, FP32 scales, BF16 convolution windows, and the reserved allocator slot. These overheads produce the measured $3.88\times$ quantization-only full state checkpoint compression rather than the ideal temporal-payload $4\times$.

E.2 END-TASK COMPOSITION PROTOCOL

For Kimi-Linear, we compare the dense mixed-precision reference with ragged DASC-NR at $W_{\max} = 16$, storing the selected temporal state in the symmetric INT8 format defined above. On a hit, the state checkpoint is dequantized once into the active FP32 temporal state before recurrence continues. The composition provides $8.11\times$ state checkpoint capacity relative to the dense mixed-precision state checkpoint. Full-attention KV remains unchanged. We use the same warmup–replay protocol, complete problem sets, replay counts, prompts, and sampling seeds as Table 2; LoCoMo is not included in this composition experiment.

F LIMITATIONS

Our evaluation focuses on hybrid architectures; applicability to purely linear-attention or state-space models remains to be studied. We have not evaluated serving at tensor-parallel degrees larger than eight, so scalability beyond TP8 remains unverified.