

EdgeRock: Strong Reasoning at Constant Memory

Huishan Ji
jih@stonehill-tech.com

Weiming Zhang
zhangwm@stonehill-tech.com

Qirui Ye
qiruiy@andrew.cmu.edu

Jinchao Xu
jinchao.xu@kaust.edu.sa

Yike Guo
yikeguo@ust.hk

Hua Xu*
huaxu@ust.hk

 github.com/rockmatrix-ai/EdgeRock

 huggingface.co/rockmatrix-ai/EdgeRock

Abstract

We present EdgeRock, a family of multimodal models that deliver strong reasoning at constant memory. Most models keep reasoning history in a key-value (KV) cache that grows with every token, consuming accelerator memory, limiting concurrent requests, and competing with perception and control on edge devices. With fixed history, memory is allocated once and never runs out, however long the model reasons. Constant memory alone is not new: bounded-state models (recurrent, state-space, and sliding-window) already achieve it, but often at the cost of reasoning, because they cannot keep the distant details that later reasoning needs. We therefore introduce *Rock Memory Anchors* (RMA): the model learns what is worth remembering and stores it as KV records in a fixed-size bank, directly readable however far back the source lies. EdgeRock-4B, our first open-source release, supports text and image reasoning and reaches 82.9% mean pass@1 on AIME 2024–2026, while the bounded-state models we evaluate reach at most 18.8%, even at 14B scale. This result is on par with leading compact models whose memory grows with reasoning. Compared with its source model, EdgeRock-4B uses up to 99.1% less KV cache and delivers up to 6.42× higher end-to-end batch throughput.

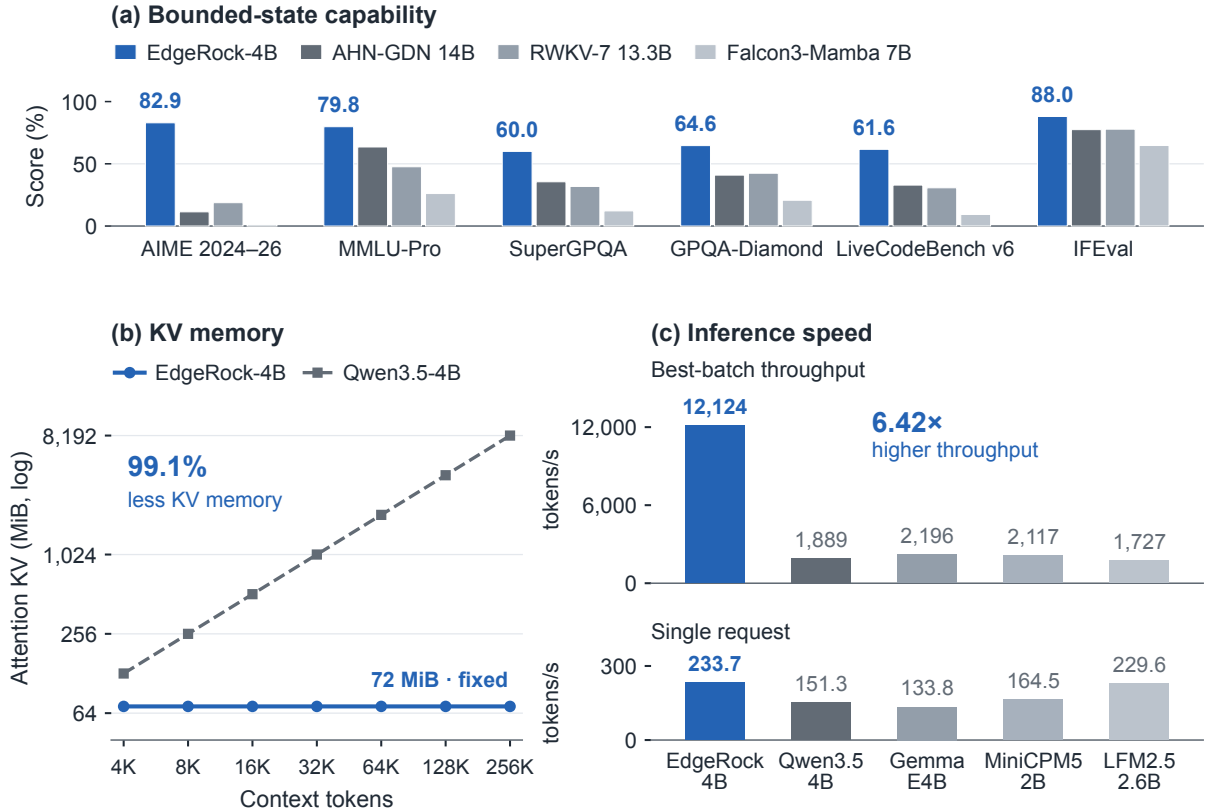


Figure 1: Reasoning at constant memory. (a) EdgeRock-4B leads the displayed 7B–14B bounded-state models on six text metrics; AIME averages 2024–2026 equally. (b) Configuration-derived BF16 attention KV capacity per sequence (log axes). EdgeRock uses 72 MiB, 99.1% less than Qwen3.5-4B at 256K. (c) Top: output-token throughput for 32K input and 32K output with repeated prompts, including prefill time, at each model’s best measured batch (Appendix Table 19); EdgeRock achieves 6.42× Qwen3.5-4B’s throughput. Bottom: cumulative decode rate for 128K input and 128K output at batch size 1. All runs use BF16 weights and KV on one Hopper 80GB GPU.

*Corresponding author.

1 Introduction

Long reasoning turns memory into the deployment bottleneck. Under full attention [1], every generated token adds to the key–value (KV) cache, so a model that reasons for 100K tokens must also store 100K tokens of history. We call such models, whose memory grows with reasoning, *growing-history* models. In data centers, their caches consume limited high-bandwidth memory (HBM), cutting the number of requests each accelerator can serve and raising serving costs [2].

For many deployments, a smaller cache is not enough: the cache must be fixed. On edge devices, robots and vehicles must reserve limited memory for perception, planning, and control [3]; dedicated chips go further and fix on-chip capacity at design time. A growing cache forces a choice between reserving space for the longest possible trace and cutting reasoning short when memory runs out. A fixed history budget makes that allocation predictable: the same memory serves a sequence at 4K or 256K tokens. What these deployments need is determinism, not only efficiency.

Recent architectures shrink this cache aggressively: multi-head latent attention (MLA) compresses each cached token [4]; hybrids such as Qwen3.5 and Kimi Linear build most layers from Gated DeltaNet (GDN) or Kimi Delta Attention (KDA) and keep full attention in only one layer in four [5, 6]; DeepSeek Sparse Attention (DSA) reads only a subset of entries [7], and the compressed sparse attention (CSA) of DeepSeek-V4 stores fewer [8]; and KV quantization methods such as KIVI use fewer bits per entry [9]. Each lowers the cost of a cached token, but the cache still grows with every token the model reads or writes. Shrinking the growth rate moves the memory wall further out; it does not remove it.

Bounded-state models already provide this determinism in history storage. Recurrent and state-space models compress history into fixed-size states [10, 11, 12], and sliding-window attention, alone or in recurrent hybrids, keeps only recent tokens [13, 14, 15]. None that we evaluate delivers usable reasoning: across 16 bounded-state models up to 14B parameters, none exceeds 18.8% on AIME 2024–2026 (Section 6). Inference-time eviction can also cap the cache of a growing-history model by keeping attention sinks, recent tokens, or tokens that drew high attention [16, 17, 18], but the model was never trained to reason with only what remains.

Long reasoning keeps returning to details stated thousands of tokens earlier: a definition, a constraint in the problem statement, an intermediate result. A sliding window forgets such a detail once it scrolls out; a recurrent state keeps only a compressed summary in which individual details blur together [19]. The question becomes: *can a model keep the distant details that later reasoning needs within a fixed memory, so that constant memory no longer costs reasoning ability?*

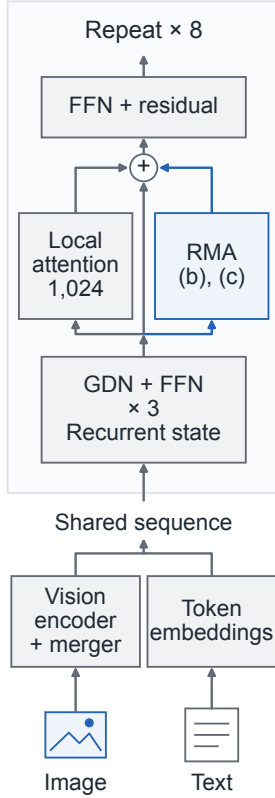
EdgeRock addresses this with *Rock Memory Anchors* (RMA). Like a rock anchor that pins a structure to bedrock, RMA pins critical details in place. Each *anchor* is a learned key–value record held in a fixed-size *anchor bank*. Recurrent state summarizes the evolving context, local attention covers the most recent 1,024 tokens, and RMA keeps the most valuable details from earlier context. A learned writer produces a candidate record for each token, and a learned scorer rates it. Once 256 candidates have accumulated, each KV group compares them with the records already in its bank, keeps the highest-scoring 1,024 records, and discards the rest. Kept records stay separate and unchanged, so any later token can attend to them directly, long after their source has left the window. Local attention and RMA together hold attention KV at 72 MiB per sequence, however long the model reasons; with recurrent state, everything a sequence must store to continue, its *persistent history*, stays at 97.76 MiB in the batch runtime.

EdgeRock-4B, our first open release, shows that constant memory and strong reasoning are compatible (Table 1).

Table 1: EdgeRock-4B at a glance.

Claim	How EdgeRock achieves it	Supporting result
Constant memory	Recurrent state, a 1,024-token window, and RMA’s fixed-size anchor banks.	72 MiB of attention KV per sequence at any length; 99.1% below Qwen3.5-4B at 256K context (Section 4).
Usable reasoning	Learned writing, admission, and retrieval, trained in stages.	82.9% on AIME 2024–2026, versus at most 18.8% for 16 bounded-state models; within the 82.2–87.8% range of leading compact models (Section 6).
RMA is essential	Anchors stay individually retrievable after leaving the window.	Disabling RMA at inference drops long-context query accuracy from 94.1% to 0.4% (Section 7).
Faster serving	Fixed history per sequence: per-token work does not grow with context, and one GPU runs more sequences at once.	512 concurrent long-generation sequences on one Hopper 80GB GPU; 4.94× Qwen3.5-4B’s throughput on average, up to 6.42× (Section 8.1).

(a) Hybrid decoder



(b) Write and select

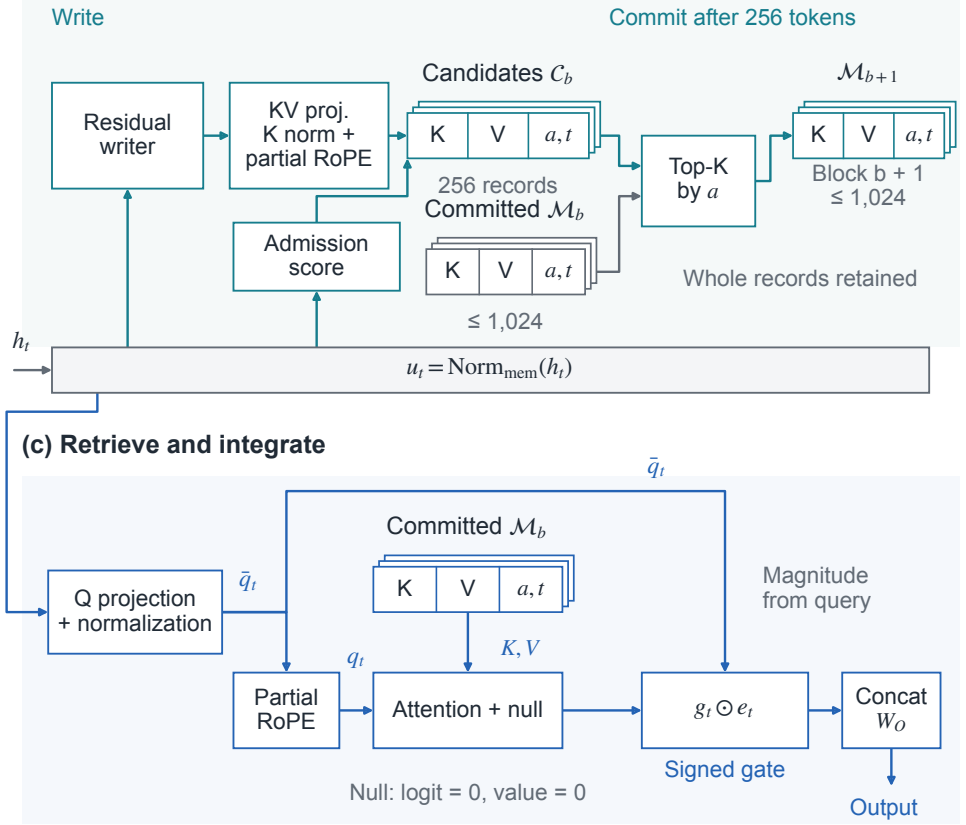


Figure 2: EdgeRock architecture. (a) Text and image features share a decoder combining recurrence, local attention, and RMA. (b) Per KV group, admission selects up to 1,024 records from the committed bank and 256 writer-generated candidates. Selections become readable in the next block. (c) Current queries read committed anchors; null attention and a query-conditioned signed gate control their contribution to the residual stream alongside local attention.

2 How is EdgeRock built?

2.1 Decoder layout

EdgeRock-4B adapts the Qwen3.5-4B language backbone [5]. It retains 24 Gated DeltaNet (GDN) layers and replaces its eight full-attention layers with a 1,024-position sliding window [13] plus an independent RMA module. The resulting pattern is

$$[\text{GDN}, \text{GDN}, \text{GDN}, \text{SWA}_{1024} \parallel \text{RMA}] \times 8. \quad (1)$$

Every sequence mixer is followed by a feed-forward sublayer; each of the eight RMA modules has its own parameters and state. Image features from the visual encoder and merger join text embeddings in a shared decoder input sequence (Figure 2a).

GDN uses fixed-dimensional recurrent matrices [20]. The local window and anchor bank retain separate KV records; a source position may appear in both.

2.2 Fixed capacity, adaptive content

EdgeRock keeps the size of its persistent history constant and updates only its contents. Let $t = 0, 1, \dots$ index causal sequence positions and $b(t) = \lfloor t/C \rfloor$ identify the current candidate block of $C = 256$ positions. Immediately before position t ,

$$\mathcal{H}_t = (\mathcal{R}_t, \mathcal{L}_t, \mathcal{M}_{b(t)}, \mathcal{C}_t, \pi_t), \quad (2)$$

$$(y_t, \mathcal{H}_{t+1}) = F_\theta(x_t, \mathbf{p}_t, \mathcal{H}_t). \quad (3)$$

Here x_t is the input embedding, y_t is the decoder output, and $\mathbf{p}_t$ contains the rotary coordinates. Across the decoder, $\mathcal{R}_t$ collects recurrent states, $\mathcal{L}_t$ local KV, $\mathcal{M}_{b(t)}$ committed anchor banks, $\mathcal{C}_t$ pending candidates, and π_t progress metadata.

2.3 Integration into the reasoning computation

Suppressing layer indices, each local-plus-RMA layer integrates retrieved information into the residual stream alongside local attention:

$$z_t = h_t + \text{SWA}(\text{Norm}_{\text{local}}(h_t)) + \text{RMA}(h_t, \mathcal{M}_{b(t)}), \quad (4)$$

$$h_t^{\text{out}} = z_t + \text{MLP}(\text{Norm}_{\text{fin}}(z_t)). \quad (5)$$

RMA uses independent normalization, writing, admission, query, key, and value projections, and output projection. The residual z_t carries local and retrieved information to later layers.

Table 2: EdgeRock-4B configuration.

Quantity	Value	Quantity	Value
Model scale	4B	RMA parameters	372M
Recurrent / local-plus-RMA layers	24 / 8	Residual width	2,560
Attention query heads / KV groups	16 / 4	Attention head dimension	256
Local window W	1,024	Committed capacity K	1,024
Candidate block C	256	RMA KV precision	BF16

2.4 Staged conversion

EdgeRock-4B is converted from Qwen3.5-4B in four stages. RMA adaptation (M0) teaches the new RMA modules to retain and retrieve information needed for later predictions. Joint adaptation (R) trains RMA together with the recurrent and local-attention backbone, so downstream layers learn to use what is retrieved. Supervised post-training (S) develops general-purpose responses with the adapted history mechanism, and reinforcement learning with verifiable rewards (RLVR) [21, 22] refines responses using feedback on answer correctness and task completion. Section 7 tracks memory accuracy after each stage (Figure 5a).

3 How does RMA preserve distant information?

At each token, RMA reads the committed anchor bank and produces a candidate record from the same causal input. At each block boundary, admission selects records from the bank and the new candidates; the updated bank becomes readable in the next block. Local attention covers recent tokens while candidates accumulate. Algorithm 1 in Appendix E gives the complete per-token procedure.

3.1 Independent memory writer

Let $u_t = \text{Norm}_{\text{mem}}(h_t) \in \mathbb{R}^D$, with $D = 2,560$. An independent residual writer with a SwiGLU-style gated projection [23] constructs a storage representation:

$$w_t = u_t + W_{\text{down}}[\text{SiLU}(W_{\text{gate}}u_t) \odot W_{\text{up}}u_t]. \quad (6)$$

RMA KV projections use the writer output w_t ; query projections use u_t . The writer has an intermediate width of 2,560 [24, 25].

3.2 Causal admission in four independent KV groups

A small network assigns a retention priority to each KV group:

$$a_t = W_{a,2} \text{SiLU}(W_{a,1}u_t), \quad a_t \in \mathbb{R}^4. \quad (7)$$

Bias-free projections of size $2560 \rightarrow 256 \rightarrow 4$ compute FP32 scores from the current causal context [24, 25]. Each group independently retains its highest-scoring records; source positions may overlap across groups.

3.3 Blockwise commits and causal visibility

Every 256 tokens, the committed bank and new candidates compete for 1,024 slots per KV group.

For block $\mathcal{B}_b = \{bC, \dots, (b+1)C - 1\}$, group g , capacity $K = 1024$ and block size $C = 256$,

$$C_{b,g} = \{(k_{t,g}, v_{t,g}, a_{t,g}, t) : t \in \mathcal{B}_b\}, \quad (8)$$

$$\mathcal{M}_{b+1,g} = \text{TopK}_{a_g}(\mathcal{M}_{b,g} \cup C_{b,g}; K). \quad (9)$$

During block b , queries read $\mathcal{M}_{b,g}$; selected candidates become readable in block $b+1$. The 1,024-position local window covers current-block interactions while the 256-token candidate buffer fills. Each update inspects at most $K + C = 1280$ records per group and then reuses the candidate buffer.

The operator $\text{TopK}(A; K)$ retains up to K records, ordered by descending score and then earlier causal position. With fixed write-time scores, the committed bank satisfies the exact history invariant

$$\mathcal{M}_{b,g} = \text{TopK}_{a_g} \left(\bigcup_{r=0}^{b-1} C_{r,g}; K \right), \quad \mathcal{M}_{0,g} = \emptyset. \quad (10)$$

Every discarded record already has at least K records ahead of it, so later candidates cannot bring it into the global top K . The update therefore maintains the highest-priority records from all completed blocks using bounded storage. Capacity limits record count, not retention age: a record remains directly addressable until displaced by admission.

3.4 Query-dependent retrieval from the anchor bank

A later query can address any retained record, including one whose source position has left the local window. Each module uses 16 query heads over four KV groups [26], with head dimension $d = 256$. Queries and keys use learned RMS normalization [27, 24, 25]:

$$\text{RMS}_\gamma(x) = \frac{\gamma \odot x}{\sqrt{d^{-1} \sum_{i=1}^d x_i^2 + 10^{-6}}}. \quad (11)$$

The query reads the current representation u_t ; keys and values store the writer representation w_t :

$$\bar{q}_{t,h} = \text{RMS}_{\gamma_Q}(W_{Q,h}u_t), \quad q_{t,h} = R(\mathbf{p}_t)\bar{q}_{t,h}, \quad (12)$$

$$k_{t,g} = R(\mathbf{p}_t) \text{RMS}_{\gamma_K}(W_{K,g}w_t), \quad v_{t,g} = W_{V,g}w_t. \quad (13)$$

Each projection $W_{Q,h}, W_{K,g}, W_{V,g} \in \mathbb{R}^{d \times D}$. Query and key gains are separate and shared across heads or groups within a module, with effective gain $\gamma = 1 + \omega$. The partial rotary transform [28] $R(\mathbf{p}_t)$ rotates 64 of 256 coordinates with base 10^7 ; values are stored without this transform or Q/K normalization. Text-only input uses $\mathbf{p}_t = (t, t, t)$, while multimodal input uses the backbone’s spatial and temporal coordinates (Appendix E).

For head h , assigned to KV group $g(h)$, let $\mathcal{I}_{b,g}$ index the records in $\mathcal{M}_{b,g}$ and define $\ell_{tj,h} = q_{t,h}^\top k_{j,g(h)}/\sqrt{d}$. With $b = b(t)$, the committed-bank read is

$$e_{t,h} = \sum_{j \in \mathcal{I}_{b,g(h)}} \frac{\exp(\ell_{tj,h})}{1 + \sum_{i \in \mathcal{I}_{b,g(h)}} \exp(\ell_{ti,h})} v_{j,g(h)}. \quad (14)$$

Each head reads at most 1,024 records. Query heads share each group’s KV bank but compute distinct attention weights.

Retrieval content and strength. The null item has zero logit and value. Its attention mass is $\alpha_{0,t,h} = (1 + Z_{t,h})^{-1}$, with $Z_{t,h} = \sum_{j \in \mathcal{I}_{b,g(h)}} \exp(\ell_{tj,h})$. For a nonempty bank, the read factors into

$$e_{t,h} = (1 - \alpha_{0,t,h}) \bar{v}_{t,h}, \quad \bar{v}_{t,h} = \sum_{j \in \mathcal{I}_{b,g(h)}} \frac{\exp(\ell_{tj,h})}{Z_{t,h}} v_{j,g(h)}. \quad (15)$$

The query determines both the retrieved content $\bar{v}_{t,h}$ and the total mass assigned to memory. The read lies in the convex hull of the stored values and zero; an empty bank returns exactly zero.

3.5 Query-conditioned memory output

The output gate acts on each retrieved head:

$$g_{t,h} = \tanh(s_h) 2\sigma \left(\frac{\bar{q}_{t,h}^\top b_h}{\sqrt{d}} + c_h \right), \quad (16)$$

$$\text{RMA}(h_t, \mathcal{M}_{b(t)}) = W_O \text{concat}_h(g_{t,h} e_{t,h}). \quad (17)$$

Here $\bar{q}$ is the normalized query before rotary encoding; $s_h, c_h \in \mathbb{R}$, $b_h \in \mathbb{R}^d$, and $W_O \in \mathbb{R}^{D \times 16d}$ are learned parameters. The gate is a signed multiplier in $(-2, 2)$. Its sign comes from the learned scale $\tanh(s_h)$; the query-conditioned sigmoid modulates its magnitude.

Admission determines which records are available, retrieval determines their content and attention mass, and the signed gate controls their contribution to the residual stream.

Relation to other retained-memory designs. Several designs also keep a small set of key–value records next to a compressed or local history. AnLLM trains a model to summarize each text segment into an anchor token and keep only those tokens’ keys and values [29]; its anchors still accumulate as the sequence grows. HOLA adds to a linear-attention model a bounded cache of recurrent keys and values ranked by a fixed, parameter-free priority [30], and LTE learns per-head scores that decide which tokens to retain as they leave a sliding window [31]. RMA differs in three ways: an independent writer produces dedicated records rather than copying existing keys and values; a causal scorer admits them into a fixed-size bank through per-group top- K commits every 256 tokens; and a separately normalized, gated read integrates them alongside local attention in a converted reasoning model.

4 What stays constant as context grows?

Fixed-size recurrent state, the local window, and anchor banks bound both stored history and per-token history access.

4.1 Memory that does not grow with sequence length

For prompt length P and generated length N , hold model dimensions, layer counts, precision, local window W , memory capacity K and block size C fixed. Storage counts fixed-width tensor elements and metadata words. The complete persistent history allocation is

$$S_{\text{history}} = S_{\text{GDN}} + S_{\text{SWA}} + S_{\text{committed}} + S_{\text{candidates}} + S_{\text{metadata}}. \quad (18)$$

Recurrent matrices, the local ring, committed anchor banks, pending candidates, scores and indices all have bounded capacity. The history allocation is consequently $O(1)$ in $P + N$ for each sequence.

$$S_{\text{committed}} = 2 \times 8 \times 4 \times 1024 \times 256 \times 2 \text{ bytes} = 32 \text{ MiB}, \quad (19)$$

$$S_{\text{candidates}} = 2 \times 8 \times 4 \times 256 \times 256 \times 2 \text{ bytes} = 8 \text{ MiB}. \quad (20)$$

The factors count keys and values, modules, KV groups, records, head width and two-byte BF16 elements. RMA uses **40 MiB of KV per sequence**. A window of 1,024 positions adds at most 32 MiB, giving **72 MiB of attention KV per sequence**. With the same eight attention layers, four KV heads, head width, and BF16 precision, a 256K-token full-attention cache requires $32 \text{ KiB} \times 262,144 = 8 \text{ GiB}$. The attention KV reduction is $1 - 72/8192 \approx 99.1\%$. Table 3 also accounts for recurrent state and auxiliary history tensors.

Table 3: Persistent history per sequence. Every entry is fixed in size, independent of context length.

Component	Storage type	MiB per sequence	Share of total
Attention KV		71.97	73.6%
Local window (1,023 preceding positions)	BF16	31.97	32.7%
RMA	BF16	40.00	40.9%
Committed anchors		32.00	32.7%
Pending candidates		8.00	8.2%
Recurrent state		25.50	26.1%
Recurrent matrices	FP16	24.00	24.6%
Convolution state	BF16	1.50	1.5%
Metadata (scores, positions, validity, counters)		0.29	0.3%
Total		97.76	100.0%

4.2 Work per generated token

Each token updates fixed recurrent states, reads at most W local positions and K anchors per group, and performs fixed-width projections and feed-forward computation, with fixed non-attention cost c_{fixed} . A block commit selects from at most $K + C$ records. Using sorting as a conservative selection bound gives

$$T_{\text{decode}}(N) = O(Nc_{\text{fixed}}) + O(N(W + K)) + O\left(\left\lceil \frac{N}{C} \right\rceil (K + C) \log(K + C)\right) = O(N). \quad (21)$$

Streamed, bounded-chunk prefill gives $O(P + N)$ **total stream-processing work**. Full attention requires $O(PN + N^2)$ attention work during decode and $O(P + N)$ history storage. Latent compression, cross-token compression, sparse reads, and hybrid layer ratios lower the constants in these terms but leave history storage linear in $P + N$; EdgeRock removes that dependence.

4.3 Inference cache and request concurrency

History storage grows with the number of concurrent requests B , not with their length. The optimized runtime implements these fixed shapes with CUDA Graphs, packed projections, fused updates, and physical sliding-window caches, and keeps each request’s history in its own slot. Requests that repeat a prompt can share its prefill, but each continuation keeps its own state.

Prefill chunks and resumed requests carry recurrent state, local KV, committed records, pending candidates, and positions together; Appendix E specifies continuation semantics.

5 Evaluation design

We compare 28 models on 29 text benchmarks and evaluate visual reasoning, distant memory, and long-sequence inference. Complete text scores appear in Appendices B and C.

5.1 Comparison objectives

The evaluation makes five comparisons, each isolating one question: 16 bounded-state models, including three 14B-class releases, show where the constant-memory class stands; the source model, Qwen3.5-4B, measures what the conversion costs; compact growing-history models, including MiniCPM5-2B and LFM2.5-2.6B, place EdgeRock-4B among models of its size; enabling and disabling RMA, together with paired changes to distant evidence, tests whether distant answers depend on the anchors; and four compact models provide the reference for decode rate, batch throughput, and GPU memory.

Bounded-state comparison class. The comparison covers recurrent and state-space models, recurrent/local hybrids [11, 32, 33, 34, 35, 36, 14, 37], and windowed attention. Phi-3 Medium 4K uses a configured 2,047-token sliding window [38].

5.2 Metrics and execution profiles

Most tasks report accuracy; DROP reports F1. Scores follow each benchmark’s prescribed subject or subtask aggregation, and the overall mean weights all 29 primary metrics equally. AIME averages the 2024–2026 editions equally. In tables that compare three or more models, bold marks the best result for each metric, including ties; paired comparisons with Qwen3.5-4B report retention instead.

Memory and visual evaluations use reference arithmetic; runtime experiments use the optimized profiles in Section 8. Appendix A specifies sampling, scoring, and source configurations.

6 Does strong reasoning survive constant memory?

EdgeRock-4B brings bounded-state reasoning to a usable level. It ranks first among 17 bounded-state models on 24 of 29 text benchmarks (Appendix B), and its AIME score lies in the range of leading compact growing-history models (Section 6.3). The bounded-state baselines differ from EdgeRock in training data and post-training as well as in architecture, so this comparison shows where the class stands rather than the cost of constant memory alone. That cost is measured against EdgeRock’s own source model, Qwen3.5-4B (Section 6.2).

6.1 A clear lead over 16 constant-memory models

EdgeRock-4B leads the evaluated bounded-state class in both the 29-benchmark mean and competition mathematics (Figure 3). The comparison includes larger GDN, RWKV, and Mamba models [35, 32, 36]. Its equal-weight 29-benchmark mean exceeds AHN-GDN Qwen2.5 14B, the strongest baseline on this aggregate, by **23.8 percentage points**.

On AIME 2024–2026, EdgeRock-4B reaches an **82.9%** mean pass@1, compared with 18.8% for RWKV-7 G1j 13.3B, the strongest bounded-state AIME comparator (Figure 3).

6.2 Retaining 96% of the source model’s reasoning

EdgeRock-4B starts from Qwen3.5-4B’s pretrained weights, so this comparison is the closest controlled measure of what constant memory costs. The cost is small: EdgeRock-4B retains **96.3%** of Qwen3.5-4B’s equal-weight AIME 2024–2026 mean pass@1 after replacing full attention with local attention and a fixed-size anchor bank. It also reaches **97.5%** on MATH-500 (Table 4).

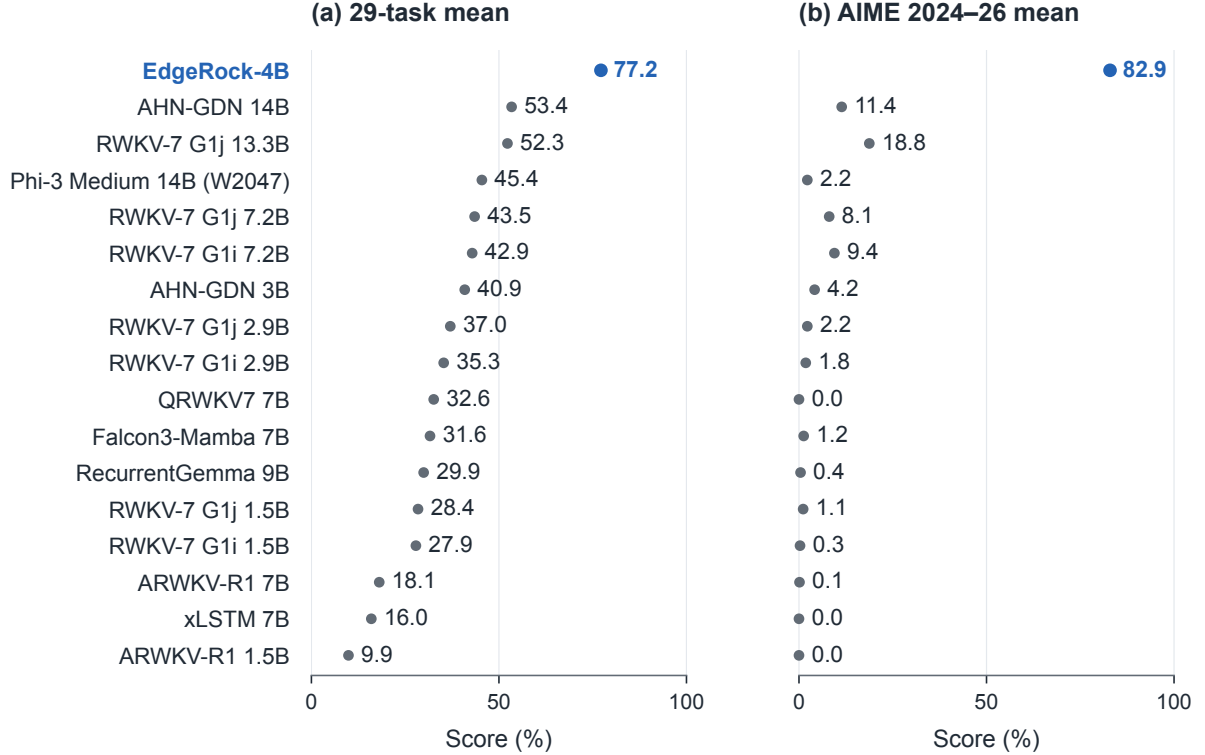


Figure 3: Bounded-state capability. (a) Equal-weight mean across 29 text benchmarks. (b) Equal-weight mean across AIME 2024–2026. Both panels order models by the 29-benchmark mean. EdgeRock-4B ranks first in both summaries among 17 models spanning 1.5B–14B. W2047 identifies Phi-3 Medium’s configured 2,047-token attention window.

Table 4: Competition mathematics and MATH-500 (mean pass@1, %). Retained gives EdgeRock-4B as a percentage of its source model, Qwen3.5-4B. AIME averages 2024–2026.

Benchmark	EdgeRock-4B	Qwen3.5-4B [5]	Retained (%)
AIME 2024 [39]	86.7	89.6	96.8
AIME 2025 [39]	77.4	80.8	95.8
AIME 2026 [39]	84.6	87.9	96.2
HMMT Feb 2025 [40]	71.8	74.6	96.2
HMMT Nov 2025 [40]	69.8	78.3	89.1
HMMT Feb 2026 [40]	63.6	65.9	96.5
MATH-500 [41, 42]	97.5	98.0	99.5
AIME three-year mean	82.9	86.1	96.3

EdgeRock-4B exceeds Qwen3.5-4B on eight of 29 text benchmarks, including MMLU, MMLU-Pro, MMLU-Redux, SuperGPQA, HumanEval, and MBPP (Figure 4).

6.3 In the range of compact models with growing memory

Against compact models whose histories grow, EdgeRock-4B reasons in the same range. On AIME 2024–2026, its 82.9% lies between Granite4.2-3B (82.2%) and Qwen3.5-4B (86.1%), with MiniCPM5-2B at 87.8%, and well above Nemotron-3-Nano-4B (57.1%), LFM2.5-2.6B (56.2%), Gemma4-E4B (40.1%), and Phi-4-mini-reasoning (38.9%) (Table 5; Appendix C). Its equal-weight mean across 29 text benchmarks is **77.2%**, compared with 76.7% for MiniCPM5-2B (Appendix C). EdgeRock-4B’s advantage over MiniCPM5-2B is larger on MMLU-Pro and SuperGPQA, which test knowledge-intensive reasoning (Table 5).

EdgeRock-4B also leads LFM2.5-2.6B on **28 of 29 benchmarks** (Appendix C).

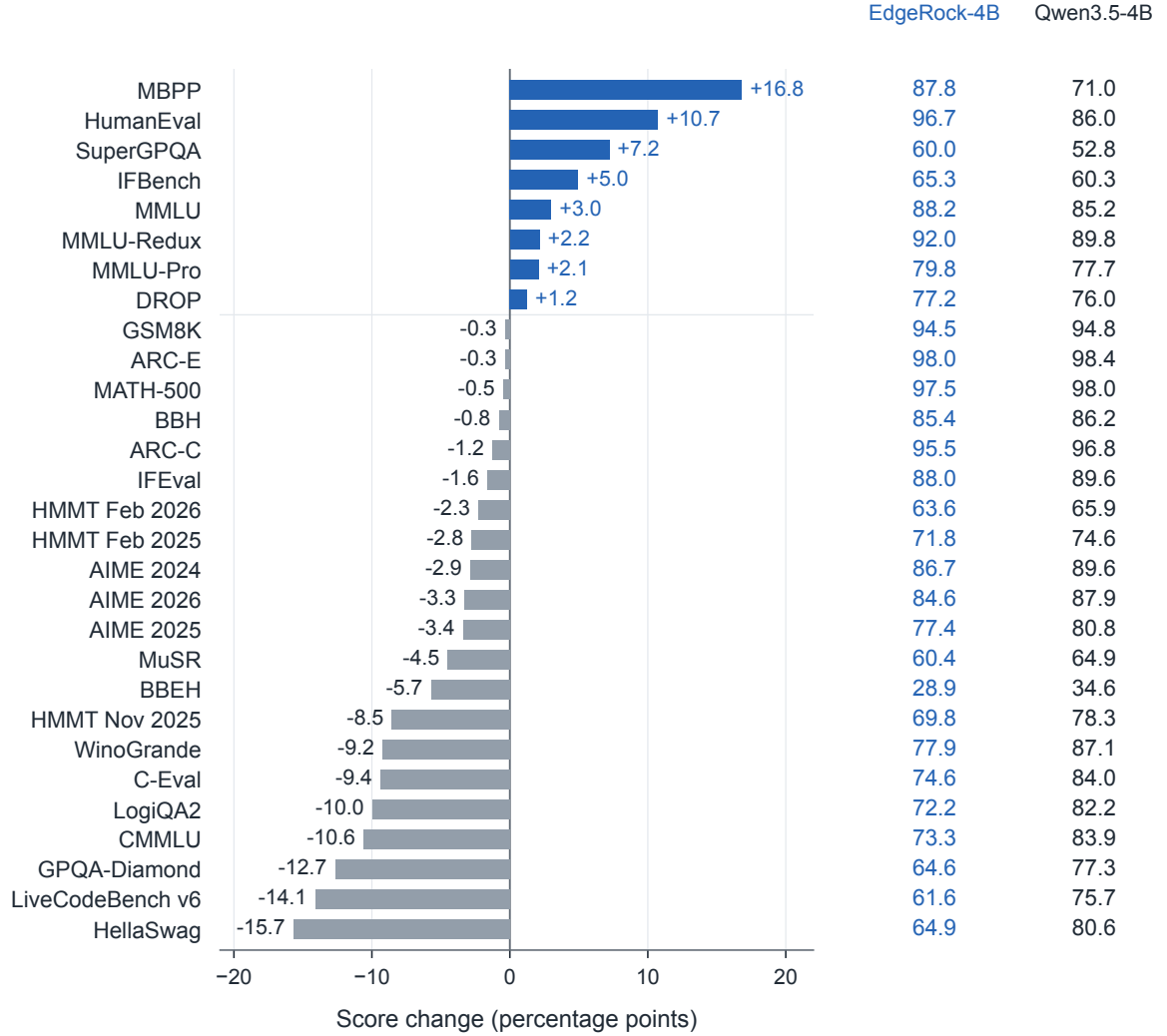


Figure 4: Capability after conversion. Bars report EdgeRock-4B minus its Qwen3.5-4B source backbone in percentage points, ordered from largest to smallest. Blue highlights the eight benchmark gains; aligned columns give absolute scores (%).

Table 5: Compact-model capability (%). EdgeRock keeps constant persistent history; models below the divider are growing-history models. AIME: 2024–2026 mean. Bold: column maximum.

Model	AIME	MMLU Pro	Super GPQA	GPQA Diamond	LCB v6	IFEval
EdgeRock-4B	82.9	79.8	60.0	64.6	61.6	88.0
Qwen3.5-4B [5]	86.1	77.7	52.8	77.3	75.7	89.6
Gemma4-E4B-it [43]	40.1	69.0	41.1	55.6	69.1	84.3
Nemotron-3-Nano-4B [44]	57.1	56.8	37.4	43.4	66.4	89.5
Phi-4-mini-reasoning [45]	38.9	38.5	32.2	48.0	35.3	45.7
MiniCPM5-2B [46]	87.8	65.3	40.9	67.7	83.3	86.3
LFM2.5-2.6B [47]	56.2	59.9	27.1	58.2	59.7	93.3

6.4 Vision retained at constant memory

EdgeRock-4B extends its constant-memory design to text-and-image reasoning. Across six visual benchmarks covering perception, chart and diagram understanding, and visual question answering, it reaches an **84.9%** equal-weight mean answer-level accuracy, compared with 89.1% for Qwen3.5-4B, retaining 95.3% (Table 6).

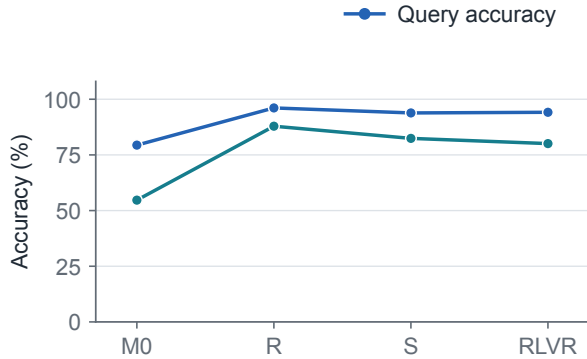
Table 6: Visual answer-level accuracy (%), including MME. Retained gives EdgeRock-4B as a percentage of Qwen3.5-4B. Appendix A gives standard MME points.

Benchmark	EdgeRock-4B	Qwen3.5-4B	Retained (%)
MMBench EN [48]	88.3	90.1	98.0
MMBench CN [48]	89.0	91.4	97.4
POPE [49]	85.4	88.8	96.2
MME [50]	83.0	86.8	95.6
ChartQA [51]	83.9	90.3	92.9
AI2D [52]	80.0	87.0	92.0
Equal-weight mean	84.9	89.1	95.3

7 Do distant answers depend on RMA?

RMA is what lets EdgeRock-4B use evidence beyond its local window. On 256 contexts of 16K to 130,000 input tokens, with four queries each covering retrieval, state updates, cross-document combination, and variable binding, it answers **94.1%** of queries; with RMA disabled, it answers **0.4%** (Figure 5). Memory is learned early and kept through post-training: query accuracy rises from 79.4% after RMA adaptation (M0) to 96.1% after joint adaptation (R) and stays at 94.1% in the final model (Figure 5a).

(a) Training progress



(b) RMA contribution

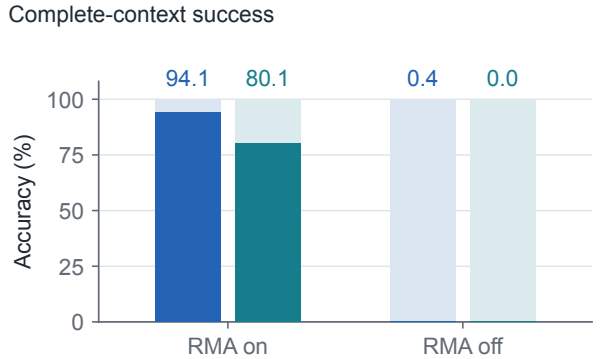


Figure 5: Distant-memory reasoning. (a) Query accuracy and complete-context success after RMA adaptation (M0), joint adaptation (R), supervised fine-tuning (S), and reinforcement learning with verifiable rewards (RLVR). Complete-context success requires all four answers to be correct. (b) The final model with all eight RMA modules on or off; recurrence and local attention remain active in both settings. With RMA enabled, EdgeRock correctly answers 94.1% of queries and completes 80.1% of contexts across retrieval, state updates, cross-document combination, and variable binding.

7.1 Without RMA, distant recall collapses

All eight RMA modules are disabled at inference time while the recurrent backbone and local attention remain active. Correct answers fall from 964 to 4 of 1,024 queries, and complete-context success from 205 to 0 of 256 contexts (Figure 5). Because RMA is disabled only at inference, this shows that in the trained model, distant answers flow through RMA rather than through recurrence or the local window.

7.2 RMA across memory tasks

Task breakdown. EdgeRock-4B achieves 90.6% complete-context success on exact retrieval, 87.5% on state-update interpretation, and 89.1% on cross-document combination (Table 7).

Paired evidence interventions. EdgeRock-4B answers both versions correctly in 89.8% of 256 pairs where changed distant evidence requires a changed answer, and in 96.5% of 256 pairs where the answer should stay the same (Table 7).

Table 7: Memory diagnostics. Top: complete-context success by task, requiring all four answers in a context to be correct (64 contexts per task). Bottom: paired changes to distant evidence, requiring both answers to be correct (256 pairs each); among pairs whose answer should stay the same, the answer never changed.

	Correct / total	Rate (%)
<i>Complete-context success by task</i>		
Exact retrieval	58 / 64	90.6
State-update interpretation	56 / 64	87.5
Cross-document combination	57 / 64	89.1
Variable binding	34 / 64	53.1
All tasks	205 / 256	80.1
<i>Paired changes to distant evidence</i>		
Answer should change	230 / 256	89.8
Answer should stay the same	247 / 256	96.5

8 What does constant memory buy in deployment?

Constant memory turns directly into serving gains. On one Hopper 80GB GPU, EdgeRock-4B decodes a single request at **233.7 tokens/s** through 128K output, runs **512** long-generation sequences at once, and averages **4.94×** Qwen3.5-4B’s end-to-end throughput. The gains come from fixed history: per-token work does not grow with context, and one GPU runs more sequences at once. Implementation details such as CUDA Graphs and fused BF16 kernels are in Appendix D.

8.1 Steady decoding and higher throughput

Single-request decoding. EdgeRock-4B sustains **233.7 tokens/s** through 128K output after 128K input, **1.54×** Qwen3.5-4B’s rate under the same workload at batch size 1. Decoding remains nearly flat throughout generation (Figure 6a). With 254K input and 2K output, reaching the same 256K total length, EdgeRock-4B decodes at **226.6 tokens/s** versus Qwen3.5-4B’s 133.9, a **1.69×** gain.

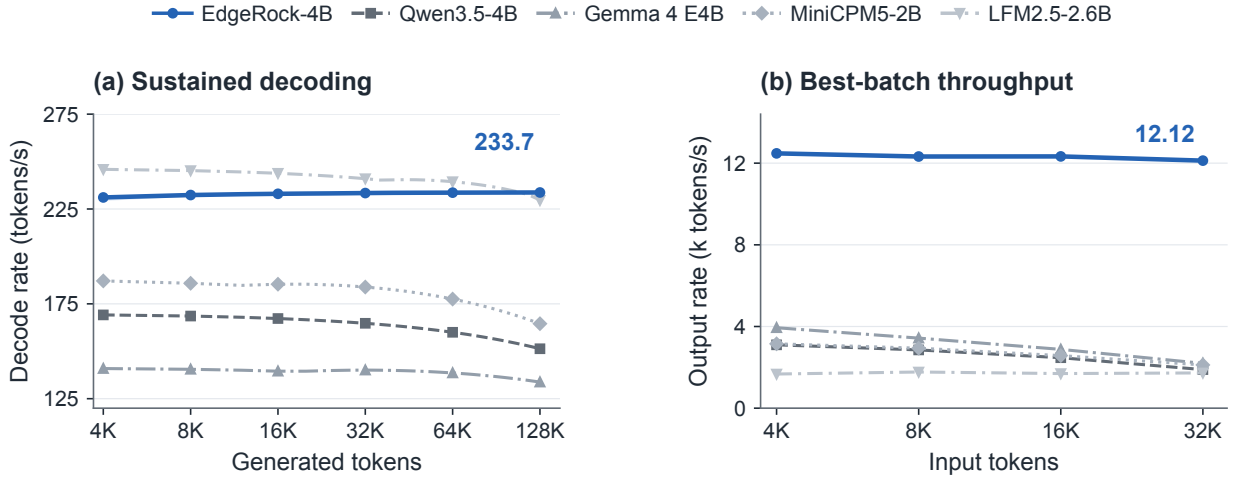


Figure 6: Long-context inference. EdgeRock-4B and four compact models use BF16 weights and KV on one Hopper 80GB GPU. (a) Cumulative decode rate during 128K output after 128K input at batch size 1. EdgeRock sustains 233.7 tokens/s through the full output. (b) Output-token throughput for 4K–32K inputs and 32K output with repeated prompts. Only output tokens are counted; elapsed time includes prefill. Each model uses its best measured batch at each input length (Appendix Table 19); EdgeRock leads across all four workloads.

Long-generation throughput. Across 4K, 8K, 16K, and 32K inputs with 32K output, EdgeRock-4B averages **4.94×** Qwen3.5-4B’s end-to-end throughput and **7.18×** LFM2.5-2.6B’s at each runtime’s best measured batch (Appendix

Table 19). The gain comes from capacity. At 32K input and 32K output, each Qwen3.5-4B sequence ends with 2 GiB of attention KV (32 KiB per token), so its cache fills after a few dozen concurrent sequences on one 80GB GPU and further requests wait. EdgeRock’s batch runtime keeps persistent history at 97.76 MiB per sequence, so all 512 sequences remain active through the full output.

8.2 Concurrency scales to 512 sequences on one GPU

EdgeRock-4B supports **512 concurrent sequences** across all 16 input/output combinations of 4K, 8K, 16K, and 32K tokens on one Hopper 80GB. At batch size 512, weighted decoding reaches **12,497 tokens/s**, or **24.4 tokens/s per active sequence** (Figure 7).

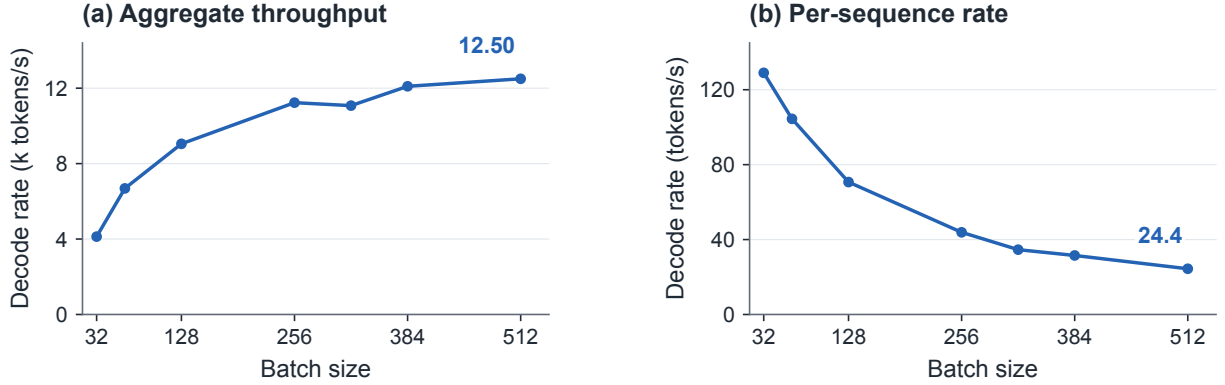


Figure 7: Concurrency scaling. EdgeRock-4B with 32–512 active sequences on one Hopper 80GB GPU (BF16 weights and KV). Rates pool decode tokens and time across all 16 input/output pairs from 4K to 32K. (a) Aggregate throughput. (b) Mean rate per active sequence.

8.3 Efficient decoding at short contexts

EdgeRock-4B reaches a decode rate of **229.6 tokens/s** at batch size 1 and **12,481 tokens/s** at batch size 512 for 4K input and 2K output, excluding prefill. At batch size 1, peak tensor allocation is **14.27 GiB** (Appendix Table 21).

8.4 Memory known before reasoning starts

In the batch runtime, each sequence carries **97.76 MiB** of history, whether EdgeRock-4B reasons for 4K tokens or 256K (Table 3). Its **72 MiB** of attention KV is **99.1%** smaller than Qwen3.5-4B’s at 256K (Figure 1b). On one Hopper 80GB GPU, a single 128K-input/128K-output request peaks at **16.02 GiB**, and 512 concurrent sequences need at most **61.26 GiB** across all 16 workloads (Figure 8).

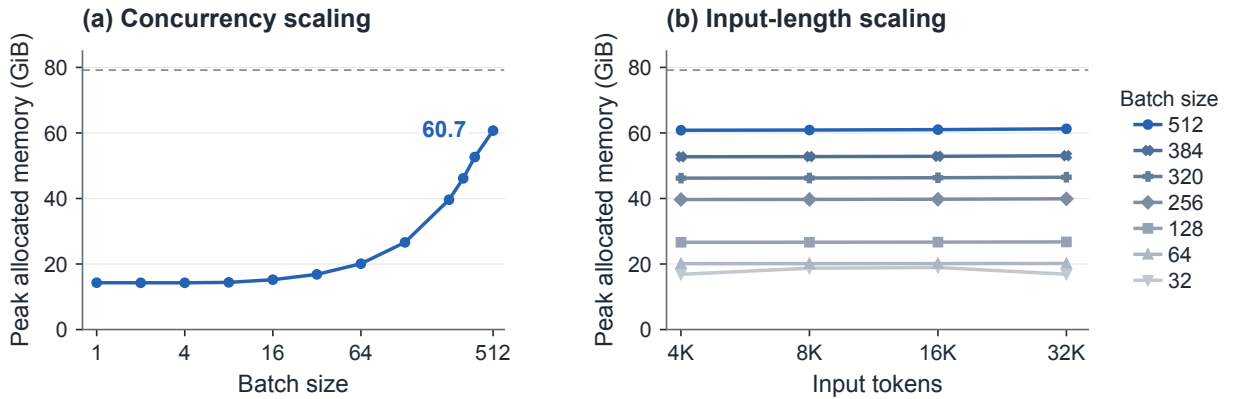


Figure 8: GPU memory scaling. Peak tensor allocation of EdgeRock-4B on one Hopper 80GB GPU, including weights, history, buffers, and workspace; the dashed line marks GPU capacity. (a) By batch size, at 4K input and 2K output. (b) By input length, with 32K output; allocation stays nearly flat at each batch size.

9 Conclusion

EdgeRock shows that strong reasoning does not require a memory that grows with every token. Once the history a model carries is fixed, memory stops being a property of the input and becomes a property of the model: it can be sized once, at design time, and relied on for as long as the model reasons.

This changes what can be built around a reasoning model. A robot can run perception and reasoning in one continuous loop without its memory budget drifting as the task goes on. A dedicated chip can hold a model’s entire history in on-chip memory, because that history no longer depends on context length. An agent’s memory no longer grows with each tool call it makes. Serving systems can plan capacity by counting requests rather than by guessing how long each will run.

Several directions follow. Scaling RMA to larger backbones, beginning with additional EdgeRock sizes now in preparation, will test whether constant memory keeps pace with stronger reasoning. Applying the same conversion to other hybrid architectures would let existing models adopt constant memory without pretraining from scratch. Anchors could extend beyond text to streams of images, audio, and sensor data, where a growing history is most costly. And because each anchor records its source position, what a model chose to remember can be inspected directly.

References

- [1] A. Vaswani et al. Attention Is All You Need. NeurIPS, 2017. arXiv:1706.03762.
- [2] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP, 2023. arXiv:2309.06180.
- [3] L. Chai et al. Accelerating LLM and VLM Inference for Automotive and Robotics with NVIDIA TensorRT Edge-LLM. NVIDIA Technical Blog, January 2026. <https://developer.nvidia.com/blog/accelerating-llm-and-vlm-inference-for-automotive-and-robotics-with-nvidia-tensorrt-edge-llm/>.
- [4] DeepSeek-AI. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. arXiv:2405.04434, 2024.
- [5] Qwen Team. Qwen3.5. Model cards: [Qwen3.5-2B](#), [Qwen3.5-4B](#), 2026.
- [6] Kimi Team. Kimi Linear: An Expressive, Efficient Attention Architecture. arXiv:2510.26692, 2025.
- [7] DeepSeek-AI. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models. arXiv:2512.02556, 2025.
- [8] DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. arXiv:2606.19348, 2026.
- [9] Z. Liu et al. KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. ICML, 2024. arXiv:2402.02750.
- [10] A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. COLM, 2024. arXiv:2312.00752.
- [11] B. Peng et al. RWKV-7 “Goose” with Expressive Dynamic State Evolution. arXiv:2503.14456, 2025.
- [12] M. Beck et al. xLSTM: Extended Long Short-Term Memory. NeurIPS, 2024. arXiv:2405.04517.
- [13] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The Long-Document Transformer. arXiv:2004.05150, 2020.
- [14] A. Botev et al. RecurrentGemma: Moving Past Transformers for Efficient Open Language Models. arXiv:2404.07839, 2024.
- [15] L. Ren et al. Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling. ICLR, 2025. arXiv:2406.07522.
- [16] G. Xiao et al. Efficient Streaming Language Models with Attention Sinks. ICLR, 2024. arXiv:2309.17453.
- [17] Z. Zhang et al. H₂O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. NeurIPS, 2023. arXiv:2306.14048.
- [18] Y. Li et al. SnapKV: LLM Knows What You Are Looking for Before Generation. NeurIPS, 2024. arXiv:2404.14469.
- [19] S. Jelassi et al. Repeat After Me: Transformers are Better than State Space Models at Copying. ICML, 2024. arXiv:2402.01032.
- [20] S. Yang, J. Kautz, and A. Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. ICLR, 2025. arXiv:2412.06464.
- [21] N. Lambert et al. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. arXiv:2411.15124, 2024.
- [22] D. Guo et al. DeepSeek-R1 Incentivizes Reasoning in LLMs through Reinforcement Learning. Nature, 645:633–638, 2025. arXiv:2501.12948.
- [23] N. Shazeer. GLU Variants Improve Transformer. arXiv:2002.05202, 2020.
- [24] Rock Matrix. EdgeRock inference and deployment code. GitHub, revision 49e4b0f8fdf0, accessed 11 October 2026. github.com/rockmatrix-ai/EdgeRock.
- [25] Rock Matrix. EdgeRock-4B model configuration. Hugging Face, revision e10f2b181881, accessed 11 October 2026. huggingface.co/rockmatrix-ai/EdgeRock.
- [26] J. Ainslie et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP, 2023. arXiv:2305.13245.
- [27] B. Zhang and R. Sennrich. Root Mean Square Layer Normalization. NeurIPS, 2019. arXiv:1910.07467.
- [28] J. Su et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. Neurocomputing, 568:127063, 2024. arXiv:2104.09864.
- [29] J. Pang et al. Anchor-based Large Language Models. Findings of ACL, 2024. arXiv:2402.07616.

- [30] W. Cui. A Hippocampus for Linear Attention: An Exact Memory for What the Recurrent State Forgets. arXiv:2607.02303, 2026.
- [31] M. He and P. N. Garner. Alleviating Forgetfulness of Linear Attention by Hybrid Sparse Attention and Contextualized Learnable Token Eviction. ICLR, 2026. arXiv:2510.20787.
- [32] RWKV Project. RWKV-7 G1 models. Model card, 2026. huggingface.co/BlinkDL/rwkv7-g1.
- [33] D. Goldstein et al. RADLADS: Rapid Attention Distillation to Linear Attention Decoders at Scale. COLM, 2025. arXiv:2505.03005.
- [34] Y. Lin et al. ARWKV: Pretrain Is Not What We Need, an RNN-Attention-Based Language Model Born from Transformer. arXiv:2501.15570, 2025. Model cards: [ARWKV-R1-1.5B](#), [ARWKV-R1-7B](#).
- [35] Y. Fang et al. Artificial Hippocampus Networks for Efficient Long-Context Modeling. ICML, 2026. arXiv:2510.07318.
- [36] Technology Innovation Institute. Falcon3-Mamba-7B-Instruct. Model card, 2024. huggingface.co/tiiuae/Falcon3-Mamba-7B-Instruct.
- [37] M. Beck et al. xLSTM 7B: A Recurrent LLM for Fast and Efficient Inference. ICML, 2025. arXiv:2503.13427.
- [38] M. Abdin et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. arXiv:2404.14219, 2024. Phi-3-medium-4k-instruct configuration: [config.json](#).
- [39] Mathematical Association of America. American Invitational Mathematics Examination (AIME), 2024–2026. <https://maa.org/maa-invitational-competitions/>.
- [40] Harvard–MIT Mathematics Tournament. HMMT February 2025, November 2025, and February 2026. <https://www.hmmt.org/www/archive/results>.
- [41] D. Hendrycks et al. Measuring Mathematical Problem Solving with the MATH Dataset. NeurIPS Datasets and Benchmarks, 2021. arXiv:2103.03874.
- [42] H. Lightman et al. Let’s Verify Step by Step. ICLR, 2024. arXiv:2305.20050.
- [43] Gemma Team. Gemma 4 Technical Report. arXiv:2607.02770, 2026. Model cards: [E2B](#), [E4B](#).
- [44] NVIDIA. NVIDIA-Nemotron-3-Nano-4B. Model card, 2026. huggingface.co/nvidia/NVIDIA-Nemotron-3-Nano-4B-BF16.
- [45] H. Xu et al. Phi-4-Mini-Reasoning: Exploring the Limits of Small Reasoning Language Models in Math. arXiv:2504.21233, 2025.
- [46] OpenBMB. MiniCPM5-2B. Model card, 2026. huggingface.co/openbmb/MiniCPM5-2B.
- [47] Liquid AI. LFM2 Technical Report. arXiv:2511.23404, 2025. LFM2.5-2.6B configuration, revision a4e00e83c097: [config.json](#).
- [48] Y. Liu et al. MMBench: Is Your Multi-modal Model an All-around Player? ECCV, 2024. arXiv:2307.06281.
- [49] Y. Li et al. Evaluating Object Hallucination in Large Vision-Language Models. EMNLP, 2023. arXiv:2305.10355.
- [50] C. Fu et al. MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models. NeurIPS Datasets and Benchmarks, 2025. arXiv:2306.13394.
- [51] A. Masry et al. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. Findings of ACL, 2022. arXiv:2203.10244.
- [52] A. Kembhavi et al. A Diagram Is Worth a Dozen Images. ECCV, 2016. arXiv:1603.07396.
- [53] K. Cobbe et al. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168, 2021.
- [54] D. Hendrycks et al. Measuring Massive Multitask Language Understanding. ICLR, 2021. arXiv:2009.03300.
- [55] Y. Wang et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. NeurIPS Datasets and Benchmarks, 2024. arXiv:2406.01574.
- [56] A. P. Gema et al. Are We Done with MMLU? NAACL, 2025. arXiv:2406.04127.
- [57] H. Li et al. CMMLU: Measuring Massive Multitask Language Understanding in Chinese. Findings of ACL, 2024. arXiv:2306.09212.
- [58] Y. Huang et al. C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models. NeurIPS Datasets and Benchmarks, 2023. arXiv:2305.08322.
- [59] D. Rein et al. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. COLM, 2024. arXiv:2311.12022.
- [60] X. Du et al. SuperGPQA: Scaling LLM Evaluation across 285 Graduate Disciplines. NeurIPS Datasets and Benchmarks, 2025. arXiv:2502.14739.
- [61] K. Sakaguchi et al. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. AAAI, 2020. arXiv:1907.10641.
- [62] R. Zellers et al. HellaSwag: Can a Machine Really Finish Your Sentence? ACL, 2019. arXiv:1905.07830.
- [63] P. Clark et al. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457, 2018.
- [64] M. Suzgun et al. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. Findings of ACL, 2023. arXiv:2210.09261.
- [65] M. Kazemi et al. BIG-Bench Extra Hard. ACL, 2025. arXiv:2502.19187.
- [66] M. Chen et al. Evaluating Large Language Models Trained on Code. arXiv:2107.03374, 2021.
- [67] J. Austin et al. Program Synthesis with Large Language Models. arXiv:2108.07732, 2021.
- [68] N. Jain et al. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. ICLR, 2025. arXiv:2403.07974. Release v6: <https://github.com/LiveCodeBench/LiveCodeBench>.

- [69] J. Zhou et al. Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911, 2023.
- [70] V. Pyatkin et al. Generalizing Verifiable Instruction Following. NeurIPS Datasets and Benchmarks, 2025. arXiv:2507.02833.
- [71] D. Dua et al. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. NAACL, 2019. arXiv:1903.00161.
- [72] Z. Sprague et al. MuSR: Testing the Limits of Chain-of-Thought with Multistep Soft Reasoning. ICLR, 2024. arXiv:2310.16049.
- [73] H. Liu et al. LogiQA 2.0: An Improved Dataset for Logical Reasoning in Natural Language Understanding. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2023. Dataset: <https://github.com/csitfun/LogiQA2.0>.
- [74] IBM. Granite 4.2. Model documentation, 2026. <https://www.ibm.com/granite/docs/models/granite4-2>. Model cards: 3B, 8B.
- [75] MiniCPM Team. MiniCPM4: Ultra-Efficient LLMs on End Devices. arXiv:2506.07900, 2025. MiniCPM4.1-8B model card: huggingface.co/openbmb/MiniCPM4.1-8B.
- [76] J. Shah et al. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. NeurIPS, 2024. arXiv:2407.08608.
- [77] T. Zadouri et al. FlashAttention-4: Algorithm and Kernel Pipelining Co-Design for Asymmetric Hardware Scaling. arXiv:2603.05451, 2026.
- [78] Z. Ye et al. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. MLSys, 2025. arXiv:2501.01005.
- [79] L. Zheng et al. SGLang: Efficient Execution of Structured Language Model Programs. NeurIPS, 2024. arXiv:2312.07104.

A Evaluation and verification protocol

Text evaluation and sampling Text evaluation uses fixed prompts, reference answers, and executable tests across 29 benchmarks.

Parameter	EdgeRock-4B text evaluation setting
Independent responses per item	$K_{\text{eval}} = 64$ for fewer than 100 questions; 16 for 100–10,000 questions; 1 above 10,000 questions
Sampling	Temperature 0.6; top- p 0.95; top- k 20; presence penalty 0
Seed / response cap	20260923 / 131,072 tokens
Stopping	EOS, token cap, or an exact 128–256-token mechanical loop repeated eight times

Binary task scores report mean pass@1 with benchmark-prescribed subject or subtask aggregation. Repetition counts vary by model and benchmark. AIME pools responses within each edition and weights the 2024–2026 editions equally.

Task-specific verification and aggregation The verifier extracts a committed answer independently of the reference, preferring the final-answer region and accepting a terminal answer in the reasoning region for semantic scoring. Conflicting or unresolved answers receive no credit. Code scores use relaxed program extraction followed by executable tests. IFEval and IFBench use strict constraint checkers, with language-detector seed 0.

BBH, BBEH and CMMLU retain their subtask or subject macro-averages; MMLU-Redux uses corrected choices. MuSR averages its three task accuracies. DROP reports token F1 and exact match, with F1 contributing to the 29-benchmark mean. LogiQA2 uses accuracy on the supplied English panel. Benchmark means assign equal weight to each benchmark, retaining its designated primary metric.

Parameter	Visual evaluation	Long-memory diagnostics
Generation	$K_{\text{eval}} = 1$; thinking enabled	Greedy $K_{\text{eval}} = 1$
Arithmetic / batch	Reference / 128	Reference / 1
Sampling	Temperature 1.0; top- p 0.95; top- k 20; presence penalty 1.5	Greedy selection
Seed	20260912	2026091707
Response cap	32,768 tokens	1,024 tokens
Input extent	65,536–4,194,304 pixels	Approximately 16K–130,000 tokens

Visual and long-memory evaluation The six reported visual benchmarks are AI2D, ChartQA, MMBench EN/CN, MME and POPE; the visual benchmark mean weights them equally. Answer-weighted accuracy, MME category points and MMBench circular accuracy remain separate metrics. EdgeRock scores 1,600.4 MME Perception points, 728.2 Cognition points, and 2,328.7 total points. MMBench EN circular accuracy is 81.6%; six-benchmark answer-weighted accuracy is 85.5%.

The memory suite has 256 contexts and 1,024 queries: 64 contexts each for exact retrieval, state-update interpretation, cross-document combination and variable binding, with four queries per context. Query accuracy measures individual answers; complete-context success requires all four answers to be correct. Paired interventions change distant evidence while holding the surrounding structure fixed, testing answers that should change or remain stable; paired accuracy requires *both* answers to be correct. The branch-disabled intervention retains the trained backbone and local window.

B Complete bounded-state benchmark results

Results cover EdgeRock-4B and 16 bounded-state comparators on 29 benchmarks. Values use each task’s primary metric on a 0–100 scale; bold marks column maxima, including ties. Appendix C reports the growing-history models.

Table 8: Bounded-state mathematics scores (%). MATH: MATH-500.

Model	GSM8K [53]	MATH [42]	AIME '24 [39]	AIME '25 [39]	AIME '26 [39]	HMMT Feb '25 [40]	HMMT Nov '25 [40]	HMMT Feb '26 [40]
EdgeRock-4B	94.5	97.5	86.7	77.4	84.6	71.8	69.8	63.6
ARWKV-R1 1.5B [34]	24.8	13.6	0.0	0.0	0.0	0.4	0.8	1.1
RWKV-7 G1i 1.5B [32]	60.3	28.2	0.0	0.4	0.4	1.7	5.0	0.0
RWKV-7 G1j 1.5B [32]	57.2	31.4	1.2	2.1	0.0	1.7	4.2	0.0
RWKV-7 G1i 2.9B [32]	71.7	43.8	1.7	1.7	2.1	1.7	5.0	0.8
RWKV-7 G1j 2.9B [32]	74.1	48.2	0.4	4.6	1.7	2.1	5.0	0.4
AHN-GDN Qwen2.5 3B [35]	83.8	66.2	7.1	2.1	3.3	2.9	6.7	4.9
ARWKV-R1 7B [34]	51.2	33.2	0.0	0.0	0.4	0.8	1.7	1.1
Falcon3-Mamba 7B Instruct [36]	61.4	41.8	1.7	0.8	1.2	0.8	5.4	2.7
QRWKV7 7B Instruct [33]	65.5	41.0	0.0	0.0	0.0	0.0	6.7	0.0
xLSTM 7B [37]	43.4	25.0	0.0	0.0	0.0	1.7	5.0	1.5
RWKV-7 G1i 7.2B [32]	85.1	68.6	8.8	11.7	7.9	2.1	5.0	3.4
RWKV-7 G1j 7.2B [32]	81.8	64.2	8.3	11.2	4.6	4.2	4.2	0.8
RecurrentGemma 9B IT [14]	51.7	22.8	0.4	0.8	0.0	1.2	5.4	1.9
RWKV-7 G1j 13.3B [32]	94.6	77.2	17.5	27.5	11.2	3.3	5.8	12.1
AHN-GDN Qwen2.5 14B [35]	93.6	80.0	13.8	10.0	10.4	7.5	10.0	4.9
Phi-3 Medium 4K Instruct [38]	85.3	49.0	3.3	3.3	0.0	6.7	2.5	3.0

Table 9: Bounded-state knowledge and science scores. Pro and Redux denote MMLU-Pro and MMLU-Redux; GPQA denotes GPQA-Diamond; Super denotes SuperGPQA.

Model	MMLU [54]	Pro [55]	Redux [56]	CMMLU [57]	C-Eval [58]	GPQA [59]	Super [60]
EdgeRock-4B	88.2	79.8	92.0	73.3	74.6	64.6	60.0
ARWKV-R1 1.5B [34]	17.4	7.5	17.9	15.6	15.2	13.1	5.8
RWKV-7 G1i 1.5B [32]	49.7	13.3	55.4	33.6	33.0	26.8	17.7
RWKV-7 G1j 1.5B [32]	49.3	16.6	53.8	35.3	33.3	26.3	16.1
RWKV-7 G1i 2.9B [32]	63.6	37.8	63.4	36.1	32.5	26.8	21.6
RWKV-7 G1j 2.9B [32]	64.7	37.9	58.9	37.9	36.5	31.8	21.8
AHN-GDN Qwen2.5 3B [35]	62.7	42.8	65.7	62.4	62.9	31.3	23.0
ARWKV-R1 7B [34]	32.3	8.8	35.2	29.0	29.0	13.1	8.8
Falcon3-Mamba 7B Instruct [36]	57.7	26.1	64.8	33.4	35.2	20.7	12.2
QRWKV7 7B Instruct [33]	64.6	23.0	34.1	57.7	56.0	8.6	15.4
xLSTM 7B [37]	19.6	4.6	37.9	16.9	19.5	12.1	5.3
RWKV-7 G1i 7.2B [32]	69.4	44.9	73.5	49.8	51.6	30.3	25.7
RWKV-7 G1j 7.2B [32]	67.5	46.2	72.7	50.8	48.9	36.9	25.3
RecurrentGemma 9B IT [14]	55.5	13.4	55.0	43.4	41.7	24.7	19.5
RWKV-7 G1j 13.3B [32]	76.7	47.6	81.8	67.4	66.3	42.4	31.8
AHN-GDN Qwen2.5 14B [35]	77.2	63.5	83.5	75.0	74.2	40.9	35.5
Phi-3 Medium 4K Instruct [38]	76.3	54.0	81.4	53.3	51.3	26.8	25.6

Table 10: Bounded-state commonsense and reasoning scores. Wino. denotes WinoGrande; Hella. denotes HellaSwag.

Model	Wino. [61]	Hella. [62]	ARC-C [63]	ARC-E [63]	BBH [64]	BBEH [65]
EdgeRock-4B	77.9	64.9	95.5	98.0	85.4	28.9
ARWKV-R1 1.5B [34]	31.4	7.4	19.6	22.3	9.8	1.6
RWKV-7 G1i 1.5B [32]	46.6	33.4	68.6	83.0	27.8	4.6
RWKV-7 G1j 1.5B [32]	49.3	24.6	66.2	80.8	31.4	5.0
RWKV-7 G1i 2.9B [32]	57.4	62.7	81.9	92.2	30.3	4.5
RWKV-7 G1j 2.9B [32]	58.1	65.9	83.3	91.6	39.5	6.5
AHN-GDN Qwen2.5 3B [35]	61.7	52.3	76.6	86.1	39.5	4.6
ARWKV-R1 7B [34]	34.4	10.0	47.4	56.8	16.6	2.4
Falcon3-Mamba 7B Instruct [36]	57.5	38.6	74.8	87.6	32.8	3.4
QRWKV7 7B Instruct [33]	61.8	48.0	86.3	95.4	37.4	1.9
xLSTM 7B [37]	28.3	7.9	25.1	28.2	12.8	3.3
RWKV-7 G1i 7.2B [32]	60.4	70.2	84.5	94.5	38.8	7.0
RWKV-7 G1j 7.2B [32]	59.4	71.7	83.7	94.5	41.4	7.7
RecurrentGemma 9B IT [14]	56.4	45.8	73.3	87.2	35.5	3.2
RWKV-7 G1j 13.3B [32]	72.1	77.4	91.8	96.3	48.7	7.7
AHN-GDN Qwen2.5 14B [35]	74.0	67.9	91.5	94.0	52.0	5.3
Phi-3 Medium 4K Instruct [38]	79.6	74.0	89.8	95.3	53.3	3.1

Table 11: Bounded-state coding and instruction following. LCB denotes LiveCodeBench.

Model	HumanEval [66]	MBPP [67]	LCB v6 [68]	IFEval [69]	IFBench [70]
EdgeRock-4B	96.7	87.8	61.6	88.0	65.3
ARWKV-R1 1.5B [34]	4.3	5.4	0.3	11.1	8.3
RWKV-7 G1i 1.5B [32]	42.1	36.6	7.2	37.0	14.7
RWKV-7 G1j 1.5B [32]	41.5	38.6	6.3	46.2	16.0
RWKV-7 G1i 2.9B [32]	58.5	49.2	11.6	43.1	17.0
RWKV-7 G1j 2.9B [32]	58.5	48.2	12.1	56.4	17.3
AHN-GDN Qwen2.5 3B [35]	67.7	49.0	14.6	58.6	26.3
ARWKV-R1 7B [34]	16.5	21.2	2.6	10.4	8.3
Falcon3-Mamba 7B Instruct [36]	34.1	39.2	9.3	64.7	18.7
QRWKV7 7B Instruct [33]	62.2	30.2	12.4	59.0	21.3
xLSTM 7B [37]	36.6	9.2	2.2	34.0	14.7
RWKV-7 G1i 7.2B [32]	65.2	59.0	18.8	53.4	17.7
RWKV-7 G1j 7.2B [32]	72.6	58.0	21.3	66.5	18.3
RecurrentGemma 9B IT [14]	37.2	19.0	7.9	39.7	12.7
RWKV-7 G1j 13.3B [32]	86.0	67.4	30.7	77.8	27.3
AHN-GDN Qwen2.5 14B [35]	75.6	68.6	32.9	77.4	33.3
Phi-3 Medium 4K Instruct [38]	73.8	59.4	19.1	61.4	22.0

Table 12: Bounded-state reading and multi-step reasoning. DROP: F1; MuSR: task-macro accuracy; LogiQA2: accuracy.

Model	DROP F1 [71]	MuSR [72]	LogiQA2 [73]
EdgeRock-4B	77.2	60.4	72.2
ARWKV-R1 1.5B [34]	5.6	8.6	17.5
RWKV-7 G1i 1.5B [32]	23.0	19.3	38.7
RWKV-7 G1j 1.5B [32]	24.9	29.1	36.1
RWKV-7 G1i 2.9B [32]	24.6	31.9	47.6
RWKV-7 G1j 2.9B [32]	30.7	34.5	45.4
AHN-GDN Qwen2.5 3B [35]	32.8	37.8	50.1
ARWKV-R1 7B [34]	11.1	15.4	26.8
Falcon3-Mamba 7B Instruct [36]	19.9	29.1	41.4
QRWKV7 7B Instruct [33]	21.2	2.6	33.2
xLSTM 7B [37]	29.0	19.3	20.0
RWKV-7 G1i 7.2B [32]	33.3	49.6	53.1
RWKV-7 G1j 7.2B [32]	34.2	48.6	56.4
RecurrentGemma 9B IT [14]	37.1	35.2	40.2
RWKV-7 G1j 13.3B [32]	51.5	51.4	66.6
AHN-GDN Qwen2.5 14B [35]	66.5	56.4	72.3
Phi-3 Medium 4K Instruct [38]	51.5	51.1	62.7

C Complete growing-history benchmark results

Results compare EdgeRock-4B with eleven growing-history models on 29 benchmarks. Bold marks column maxima, including ties. Metrics and sampling follow Appendix A.

Table 13: Mathematics comparison with growing-history models (%). MATH: MATH-500.

Model	GSM8K [53]	MATH [42]	AIME '24 [39]	AIME '25 [39]	AIME '26 [39]	HMMT Feb '25 [40]	HMMT Nov '25 [40]	HMMT Feb '26 [40]
EdgeRock-4B	94.5	97.5	86.7	77.4	84.6	71.8	69.8	63.6
Qwen3.5-4B [5]	94.8	98.0	89.6	80.8	87.9	74.6	78.3	65.9
Gemma4-E2B-it [43]	90.1	78.8	38.3	33.8	33.8	21.7	17.5	20.1
MiniCPM5-2B [46]	93.6	97.0	87.5	85.4	90.4	76.7	74.6	63.3
LFM2.5-2.6B [47]	91.0	92.7	63.4	56.3	48.9	34.7	42.9	32.9
Qwen3.5-2B [5]	87.6	86.8	41.2	34.2	35.4	26.2	23.3	25.4
Granite4.2-3B [74]	93.8	97.8	85.0	78.8	82.9	75.8	72.5	59.5
Phi-4-mini-reasoning [45]	93.6	91.6	45.4	32.5	38.8	25.0	28.7	24.6
Gemma4-E4B-it [43]	91.3	88.2	42.1	40.0	38.3	25.4	32.9	28.8
Nemotron-3-Nano-4B [44]	92.4	94.8	60.8	56.7	53.8	40.0	47.1	36.4
Granite4.2-8B [74]	94.4	98.0	88.3	87.9	91.7	81.7	86.7	73.9
MiniCPM4.1-8B [75]	93.6	96.4	77.1	70.0	70.8	58.3	59.6	49.2

Table 14: Knowledge and science comparison with growing-history models. Pro and Redux denote MMLU-Pro and MMLU-Redux; GPQA denotes GPQA-Diamond; Super denotes SuperGPQA.

Model	MMLU [54]	Pro [55]	Redux [56]	CMMLU [57]	C-Eval [58]	GPQA [59]	Super [60]
EdgeRock-4B	88.2	79.8	92.0	73.3	74.6	64.6	60.0
Qwen3.5-4B [5]	85.2	77.7	89.8	83.9	84.0	77.3	52.8
Gemma4-E2B-it [43]	72.2	47.0	78.2	60.2	61.2	46.5	34.1
MiniCPM5-2B [46]	79.6	65.3	85.3	79.2	78.3	67.7	40.9
LFM2.5-2.6B [47]	75.3	59.9	81.7	57.2	59.5	58.2	27.1
Qwen3.5-2B [5]	72.5	62.8	79.5	68.2	68.7	54.5	39.3
Granite4.2-3B [74]	77.2	66.4	81.6	59.1	60.8	58.6	39.5
Phi-4-mini-reasoning [45]	77.5	38.5	78.3	45.3	42.4	48.0	32.2
Gemma4-E4B-it [43]	81.2	69.0	86.4	68.7	68.6	55.6	41.1
Nemotron-3-Nano-4B [44]	75.8	56.8	81.5	54.6	56.4	43.4	37.4
Granite4.2-8B [74]	78.6	72.0	86.3	55.1	58.4	61.6	45.7
MiniCPM4.1-8B [75]	82.7	67.0	87.8	80.2	79.0	51.5	40.6

Table 15: Commonsense and reasoning comparison with growing-history models. Wino. denotes WinoGrande; Hella. denotes HellaSwag.

Model	Wino. [61]	Hella. [62]	ARC-C [63]	ARC-E [63]	BBH [64]	BBEH [65]
EdgeRock-4B	77.9	64.9	95.5	98.0	85.4	28.9
Qwen3.5-4B [5]	87.1	80.6	96.8	98.4	86.2	34.6
Gemma4-E2B-it [43]	63.2	50.9	89.2	94.8	80.3	11.6
MiniCPM5-2B [46]	69.8	52.2	91.9	97.0	82.1	25.4
LFM2.5-2.6B [47]	67.9	40.7	91.6	96.5	76.7	13.1
Qwen3.5-2B [5]	71.5	59.3	91.4	95.5	79.1	14.9
Granite4.2-3B [74]	74.0	67.4	92.4	97.1	85.5	24.5
Phi-4-mini-reasoning [45]	75.8	62.6	90.2	94.4	72.5	12.9
Gemma4-E4B-it [43]	76.5	53.5	94.3	97.7	84.1	17.4
Nemotron-3-Nano-4B [44]	65.6	46.3	91.9	96.7	79.3	10.8
Granite4.2-8B [74]	76.2	63.1	90.9	94.6	88.9	31.5
MiniCPM4.1-8B [75]	78.2	49.8	93.4	97.4	86.6	14.0

Table 16: Coding and instruction following comparison with growing-history models. LCB denotes LiveCodeBench.

Model	HumanEval [66]	MBPP [67]	LCB v6 [68]	IFEval [69]	IFBench [70]
EdgeRock-4B	96.7	87.8	61.6	88.0	65.3
Qwen3.5-4B [5]	86.0	71.0	75.7	89.6	60.3
Gemma4-E2B-it [43]	90.9	79.6	59.6	82.4	29.0
MiniCPM5-2B [46]	97.6	94.8	83.3	86.3	64.0
LFM2.5-2.6B [47]	92.9	73.6	59.7	93.3	56.2
Qwen3.5-2B [5]	54.9	35.4	25.0	81.5	43.7
Granite4.2-3B [74]	97.0	92.6	75.9	92.2	74.3
Phi-4-mini-reasoning [45]	79.9	64.0	35.3	45.7	14.3
Gemma4-E4B-it [43]	93.9	79.2	69.1	84.3	34.7
Nemotron-3-Nano-4B [44]	90.9	72.2	66.4	89.5	57.7
Granite4.2-8B [74]	96.3	95.4	85.6	94.6	77.0
MiniCPM4.1-8B [75]	98.2	92.2	70.4	73.4	23.7

Table 17: Reading and multi-step reasoning comparison with growing-history models. DROP: F1; MuSR: task-macro accuracy; LogiQA2: accuracy.

Model	DROP F1 [71]	MuSR [72]	LogiQA2 [73]
EdgeRock-4B	77.2	60.4	72.2
Qwen3.5-4B [5]	76.0	64.9	82.2
Gemma4-E2B-it [43]	72.9	58.9	59.4
MiniCPM5-2B [46]	76.0	65.3	73.7
LFM2.5-2.6B [47]	72.3	60.4	63.2
Qwen3.5-2B [5]	73.7	61.3	66.3
Granite4.2-3B [74]	83.7	65.1	69.5
Phi-4-mini-reasoning [45]	50.6	56.7	58.5
Gemma4-E4B-it [43]	72.9	62.2	72.6
Nemotron-3-Nano-4B [44]	72.6	58.2	67.9
Granite4.2-8B [74]	76.9	62.4	73.8
MiniCPM4.1-8B [75]	78.3	64.5	75.7

D Runtime protocols and memory accounting

D.1 Workloads and hardware

All runs use one NVIDIA Hopper 80GB GPU and tensor-parallel size one. All matrix weights and KV are BF16.

Here 1K = 1,024 tokens. The sustained single-request workload (batch size = 1) is 128K input plus 128K output. A second single-request workload uses 254K input and 2K output, with three independent runs per model for EdgeRock and Qwen; both workloads total 262,144 tokens. Long-generation throughput uses 4K/8K/16K/32K inputs and 32K output. The full EdgeRock sweep combines those four input lengths with the same four output lengths at batches 32, 64, 128, 256, 320, 384, and 512. A separate 4K-input/2K-output sweep covers batch sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 320, 384, and 512. Each EdgeRock configuration uses three independent runs; rates are medians unless specified as weighted.

The benchmark uses synthetic input tokens, EOS ignored, and a 256-token warmup at the target input length and batch capacity. EdgeRock uses full-vocabulary autoregressive sampling with temperature 1.0, top- k 20, top- p 0.95, and presence penalty 1.5.

D.2 Timed intervals and aggregation

Decode rate measures output after each request’s first token, including sampling, state updates, graph management, and runtime scheduling. The single-request curves report cumulative rate every 4K output tokens. **End-to-end output-token throughput** divides the total number of generated output tokens by the full generation-call duration, including prefill time. Input tokens do not enter the token count. Both exclude model loading, warmup, tokenization, and text decoding.

Mean throughput ratios weight the four 32K-output workloads equally, using each runtime’s best measured batch. Weighted batch throughput divides total subsequent output tokens by total decode time across all 16 length pairs and three runs; per-sequence rate divides it by the active batch. Bands show the range across runs. Memory results use peak allocated and reserved memory across runs.

D.3 Runtime configuration and prompt reuse

Packed projections and fused normalization, rotary encoding, and memory operations reduce data movement. At batch size 512, the runtime shares original and packed BF16 weight storage, saving 4.95 GiB. At batch size 1, the runtime fuses output-head processing with full-vocabulary sampling and state updates. Decoding is autoregressive, with weights and history on the GPU.

EdgeRock and LFM test batches 32, 64, 128, 256, 320, 384, and 512; Qwen and MiniCPM test 32 and 64; Gemma tests 64 and 128. Batch denotes submitted requests. EdgeRock maintains the target concurrency; vLLM schedules active requests within its cache budget. Eight continuations share each prompt in the EdgeRock batch grid, then retain independent recurrent, local-attention, and anchor state. The short sweep shares a prompt across up to eight requests.

The vLLM peers use CUDA Graphs, chunked prefill, and a 16K prefill-token budget. Qwen, Gemma, and MiniCPM use asynchronous scheduling. Gemma, MiniCPM, and LFM use 95% GPU-memory utilization. The synthetic single-request workloads use a 256K execution capacity with unchanged weights and RoPE frequencies. Gemma E4B denotes effective scale; its full checkpoint is approximately 8B parameters. MiniCPM’s checkpoint is approximately 2.52B.

Table 18: Runtime configuration of each model on one Hopper 80GB GPU.

Model	History precision	Execution path
EdgeRock-4B	BF16 KV; GDN: FP32 at batch size 1, FP16 otherwise; FP32 updates	Custom CUDA Graph engine; fused BF16 kernels; directly indexed state and local rings
Qwen3.5-4B	BF16 KV; FP32 GDN state	vLLM [2] CUDA Graphs; FlashAttention-3 [76] (batch size 1), -4 [77] (batch); FlashInfer [78] GDN prefill and fused decode
Gemma 4 E4B	BF16 KV	vLLM O3; FlashAttention-4; language model only
MiniCPM5-2B	BF16 KV	vLLM O3; FlashAttention-3
LFM2.5-2.6B	BF16 KV	vLLM 0.23.0; CUDA Graphs and chunked prefill

D.4 History and device allocation

The 72 MiB of attention KV is a configuration-derived bound. During decoding, the local cache stores 1,023 preceding positions and the decoder supplies the current token. The history tensors in Table 3 total 97.76 MiB per sequence in the batch runtime, including recurrent state, convolution state, and anchor metadata.

Peak tensor allocation includes weights, history, input/output buffers, graph storage, and temporary workspace. Reserved memory also includes cached allocator blocks. At batch size 512, shared storage for original and packed BF16 projections eliminates duplicate weights while retaining the full history capacity.

D.5 Measured throughput and memory

Tables 19–21 list the measurements behind Figures 6–8.

Table 19: Output-token throughput with 32K output on one Hopper 80GB. Only output tokens are counted; elapsed time includes prefill. Columns give input length; cells show output tokens/s (best batch size). Bold marks the highest throughput in each column.

Model	4K	8K	16K	32K
EdgeRock-4B	12,479 (512)	12,325 (512)	12,330 (512)	12,124 (512)
Qwen3.5-4B	3,103 (64)	2,853 (64)	2,472 (64)	1,889 (64)
Gemma 4 E4B	3,940 (128)	3,432 (128)	2,880 (128)	2,196 (64)
MiniCPM5-2B	3,152 (64)	2,938 (64)	2,575 (32)	2,117 (32)
LFM2.5-2.6B	1,673 (64)	1,771 (64)	1,695 (32)	1,727 (64)

Table 20: Weighted decoding and peak GPU memory across 16 input/output pairs (4K–32K) on one Hopper 80GB.

Batch size	Decode, tokens/s	Per sequence, tokens/s	Allocated, GiB	Reserved, GiB
32	4,128	129.0	18.94	19.49
64	6,685	104.5	20.16	21.80
128	9,049	70.7	26.74	27.92
256	11,231	43.9	39.90	41.77
320	11,074	34.6	46.49	50.69
384	12,099	31.5	53.06	57.27
512	12,497	24.4	61.26	66.97

Table 21: Decode throughput and peak GPU allocation at 4K input and 2K output on one Hopper 80GB.

Batch size	Decode, tokens/s	Per sequence, tokens/s	Allocated, GiB
1	230	229.6	14.27
2	358	178.8	14.27
4	692	172.9	14.27
8	1,322	165.2	14.40
16	2,442	152.6	15.21
32	4,053	126.7	16.84
64	6,698	104.7	20.10
128	9,066	70.8	26.60
256	11,100	43.4	39.62
320	11,062	34.6	46.14
384	12,086	31.5	52.64
512	12,481	24.4	60.71

E Streaming execution and state ownership

E.1 State owned by a continuing request

A continuing request owns recurrent matrices, a local KV ring, committed anchor banks, pending candidates, and progress metadata (Table 22). This state persists across pauses and slot transfers. New anchors become readable after the next block commit.

Table 22: Logical state needed to continue one request. Capacities are per RMA module and KV group where applicable.

State	Contents	Maintenance rule
Recurrent state	The GDN history representation	Update fixed-dimensional recurrent matrices.
Local history	Recent attention KV and the ring position	Retain at most $W = 1,024$ positions.
Committed anchor bank	Selected KV records, admission scores, and source positions	Retain at most $K = 1,024$ records per group.
Pending candidates	Current-block records, scores, and occupancy	Accumulate up to $C = 256$ positions before a commit.
Progress metadata	Position and block progress, plus association with the active request	Preserve consistent continuation and slot ownership.

Model weights are shared across requests; history occupies independent slots in shared cache pools. Extending a request reuses its history capacity, while admitting another request allocates an additional slot.

E.2 Forward procedure and causal visibility

Prefill and decode follow Algorithm 1 with the same logical commit boundaries. Queries read the committed bank throughout each 256-token block, while local attention covers recent tokens. A boundary commit makes selected candidates available to the next block.

Algorithm 1 Causal RMA update in one local-plus-RMA layer

Require: Hidden input h_t , causal index t , rotary coordinates $\mathbf{p}_t$

Require: Local state $\mathcal{L}$, committed anchor bank $\mathcal{M}$, pending candidates C

Require: Window $W = 1,024$, bank capacity $K = 1,024$, candidate block size $C = 256$

```

1:  $u_t \leftarrow \text{Norm}_{\text{mem}}(h_t)$ 
2:  $m_t \leftarrow \text{ReadAndGate}(u_t, \mathbf{p}_t, \mathcal{M})$  ▷ Committed bank only
3:  $w_t \leftarrow \text{Writer}(u_t); (k_t, v_t) \leftarrow \text{MemoryKV}(w_t, \mathbf{p}_t)$ 
4:  $a_t \leftarrow \text{Admission}(u_t)$  ▷ Four causal group scores
5:  $(\ell_t, \mathcal{L}) \leftarrow \text{LocalAttentionStep}(h_t, \mathbf{p}_t, \mathcal{L}; W)$ 
6:  $z_t \leftarrow h_t + \ell_t + m_t; h_t^{\text{out}} \leftarrow z_t + \text{MLP}(\text{Norm}_{\text{ffn}}(z_t))$ 
7: Append  $(k_t, v_t, a_t, t)$  to  $C$ 
8: if  $|C| = C$  then ▷ Logical block boundary, independent of prefill chunks
9:   for  $g = 1, \dots, 4$  do
10:     $M_g \leftarrow \text{TopK}_{a_g}(M_g \cup C_g; K)$ 
11:   end for
12:    $C \leftarrow \emptyset$  ▷ Updated bank is visible to the next block
13: end if
14: return  $h_t^{\text{out}}, \mathcal{L}, \mathcal{M}, C$ 

```

`ReadAndGate` combines query projection, committed-bank attention with the null item, the signed gate, and output projection (Section 3). `MemoryKV` applies the writer’s key and value projections, key normalization, and rotary encoding. Local attention includes the current token and retains at most $W = 1,024$ positions. Top- K ties favor earlier source positions. A new sequence starts with empty local state, anchor banks, and candidate buffers; unfinished candidate blocks persist across prefill chunks and request pauses.

E.3 Selection invariants and information availability

Each KV group examines at most 1,280 records at a boundary and retains up to 1,024, preserving the surviving key and value vectors unchanged. Fixed write-time scores and consistent tie ordering maintain the history top- K invariant in Equation 10. Query heads share the group’s bank and compute independent retrieval weights. Local and memory attention normalize separately; an empty bank returns zero.

E.4 Continuation, chunking, and request lifecycle

The portable backend defaults to 1,024-token prefill chunks. Chunking bounds the working set while preserving local, recurrent, and anchor state. RMA commits remain tied to completed candidate blocks, independently of execution chunk boundaries.

Prefix resumption restores every component in Table 22, including unfinished candidates, so their records participate in the next commit.

Native adapters access history in place through GPU request-slot indices. SGLang [79] reset and copy operations initialize reassigned slots and preserve continuing requests.

Released positional configuration. RMA uses partial RoPE over 64 of 256 coordinates, base 10^7 , default scaling, and a maximum-position execution setting of 262,144. Multimodal positions use interleaved rotary sections (11, 11, 10); text supplies the same cumulative token index on all three axes [24, 25].