LONGSTRAW: MEMORY-CONSTRAINED POLICY UPDATES WITH MILLION-TOKEN PREFIXES

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy training on long histories remains memory intensive even with low-rank adapters, because sequence state and activations grow with the input. We introduce LongStraw, a framework for memory-constrained policy updates on complete long inputs. It couples a compact, versioned prefix-state data structure with a capture–replay–recapture algorithm. The structure organizes layer-specific continuation tensors with their ownership, positions and gradient boundary, sharing prompt state across responses while keeping their extensions private. Capture constructs this state without autograd; block-wise reverse replay differentiates responses and an optional short prompt tail; recapture refreshes the state after each optimizer step. This bounds live activation memory while preserving access to the complete input. The update optimizes a conditional objective that omits gradients through construction of the captured prefix. On one H200, LongStraw completes million-token Qwen3.6-27B updates where native CPU-offloaded autograd of the same objective runs out of memory. On a fixed million-token question-answering benchmark, exact-match accuracy improves by 2.03 percentage points across paired training seeds. Experiments show learning gains on complete inputs up to 4M tokens, and distributed GLM-5.2 experiments demonstrate online policy updates. These results show how jointly organizing prefix state, backward computation and state refresh makes policy training on complete long histories feasible under memory constraints.

1 INTRODUCTION

Personalized assistants and interactive agents act on histories accumulated across conversations, documents and tool observations (Wu et al., 2025; Yao et al., 2023). LongMemEval includes histories up to 1.5 million tokens; LongBench v2 evaluates long-document and repository reasoning (Wu et al., 2025; Bai et al., 2025). Long-input instruction tuning and reinforcement learning improve context use (Bai et al., 2024; Wan et al., 2025; Wang et al., 2025), motivating post-training where the accumulated history dwarfs the next response or action.

Under a fixed GPU budget, peak memory determines feasible updates. Low-rank adaptation (LoRA) and quantization reduce parameter storage (Hu et al., 2022; Dettmers et al., 2023), but training still retains or reconstructs sequence activations alongside the state used to process the history (Li et al., 2026b). These costs remain even when few parameters are trainable.

We study grouped policy updates with $P \gg R$ and $G > 1$: a shared prompt of P tokens, responses of length R, and G response gradients per optimizer step (Shao et al., 2024; Yu et al., 2025; Sheng et al., 2025). Shorter inputs reduce storage; shared-prefix schedules reuse computation with full-prompt gradients (Liu et al., 2025b; Li et al., 2026a); sequence parallelism spreads computation across devices (Liu et al., 2024; Jacobs et al., 2023).

We introduce LongStraw, coupling a versioned prefix-state structure with a capture–replay–recapture algorithm. Compact per-layer state replaces the prompt computation graph and supports private response extensions. Replay differentiates responses and an optional short prompt tail; recapture refreshes state after parameter updates (Figure 1). Every response conditions on the complete input, while the conditional objective omits prefix-construction gradients.

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

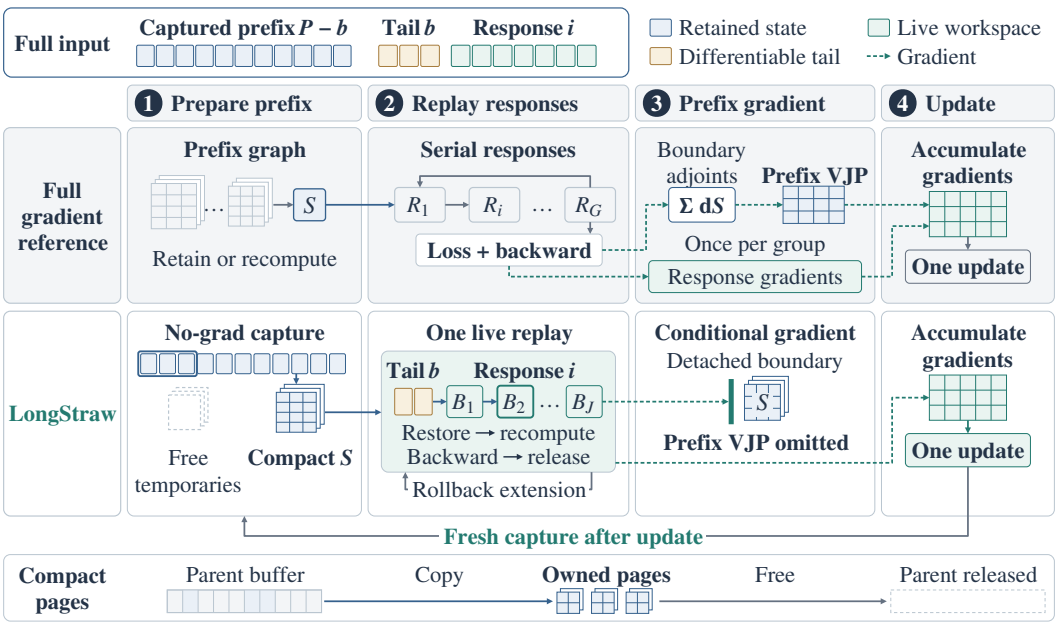


Figure 1: **LongStraw keeps prompt state but no prefix graph.** The full-gradient reference retains or recomputes a prefix graph and runs one prefix vector–Jacobian product (VJP) (Liu et al., 2025b; Li et al., 2026a). LongStraw detaches the earlier prefix, rebuilds response activations and may differentiate a b-token tail. Both read the full input and take one optimizer step but optimize different objectives. $R_1, \ldots, R_G$ are the responses and $B_1, \ldots, B_J$ the replay blocks of response i. Dashed gradient arrows stop at the detached boundary. Inset: compact pages free the parent capture buffer.

The structure and algorithm address coupled costs: retained views can keep capture buffers alive even without autograd, while backward traverses recurrent, attention and expert state. Compact storage releases capture buffers, replay bounds live computation, and versioning keeps shared state consistent with the updated policy. We make three contributions:

- **A training-state structure for shared long histories.** We organize compact layer-specific tensors with ownership, positions, a gradient boundary and a parameter version. Shared prefixes and private extensions support grouped updates across hybrid attention and mixture-of-experts architectures (Sections 4.1 and 4.3).
- **A capture–replay–recapture update algorithm.** LongStraw propagates derivatives across tail and response blocks, shares prompt state within an update and rebuilds it after parameter changes. This enables Qwen3.6-27B updates at 1M on one H200, where native CPU-offloaded autograd of the same objective runs out of memory. At 128K–512K, it also reduces update time and GPU and host memory relative to that baseline (Section 5.2).
- **Learning from complete million-token inputs.** Qwen3.6-27B gains 2.03 exact-match (EM) points at 1M across paired seeds. Single-seed studies show gains at 2M and 4M (Sections 5.3 and 5.4). Matched 8K ablations show that a 32-token prompt tail (Tail32) retains much of the learning gain of full-prompt training (Section 5.1). Distributed GLM-5.2 experiments demonstrate five online policy updates at 2M (Section 5.5).

2 Related Work

Shared-prefix training with full gradients. Prefix Grouper shares one prompt forward pass across the response group in group relative policy optimization (GRPO) and keeps full differentiability (Liu et al., 2025b). Schedule-level reuse accumulates suffix key/value (K/V) gradients (dK/dV) between one prefix forward and one backward pass, with offload and distributed support (Li et al., 2026a). These approaches share prefix computation while preserving its gradients. LongStraw studies the conditional update obtained by stopping those gradients at the prompt boundary. A differentiable tail moves this boundary inside the prompt.

Truncated gradients and cached state. Truncated backpropagation through time stops gradients after a fixed recurrent window (Williams & Peng, 1990), and Transformer-XL attends to cached earlier segments under a stop-gradient (Dai et al., 2019). In both, every scored token is predicted from a state inside the differentiated segment. In a policy update, the first answer token is predicted from the last prompt state, so a boundary at the end of the prompt silently removes that token's direct gradient (Proposition 1). LongStraw moves the boundary b tokens into the prompt, restoring this direct path, and recaptures prompt state after every optimizer step.

Long-context reinforcement learning. QwenLong-L1 applies reinforcement learning (RL) to long-document reasoning with progressive context scaling (Wan et al., 2025); LoongRL trains on synthetic 16K multi-hop tasks that transfer to 128K contexts (Wang et al., 2025); MemAgent reads chunks with RL-trained memory, using an 8K window and 32K training texts to generalize to 3.5M-token QA (Yu et al., 2026). These methods shape the training data or input format. LongStraw complements them by making the policy update itself feasible on the complete input.

Efficient long-context training and execution. Prior work studies long-context instruction tuning (Bai et al., 2024) and context extension to 4M tokens (Xu et al., 2025). LongLoRA (Chen et al., 2024), QLoRA (Dettmers et al., 2023), FlashAttention (Dao et al., 2022; Dao, 2023) and checkpointing (Chen et al., 2016) reduce attention, weight and activation memory. Ring Attention, DeepSpeed-Ulysses, DistFlashAttn and ByteScale distribute long sequences across devices (Liu et al., 2024; Jacobs et al., 2023; Li et al., 2024; Ge et al., 2025). FPDT pipelines chunks through host memory (Yao et al., 2025), SeCO serializes chunks (Li et al., 2025) and OOMB combines chunk recurrence, paged K/V and gradient storage, sparse attention and CPU offload (Li et al., 2026b); all three keep full-sequence gradients. PagedAttention pages the serving K/V cache (Kwon et al., 2023). LongStraw combines paged state and checkpointing with response replay. We study how restricting prompt gradients changes memory use and learning when the complete input is retained.

3 Training under a GPU Budget

We study policy training on a fixed set of GPUs, where parameters, optimizer state, cached prefix state and transient computation compete for device memory; the goal is to extend the context length at which a complete update fits. Host memory used by CPU offload is reported separately.

3.1 Response-Decomposable Policy Updates

Many policy-gradient objectives are weighted sums of per-response losses. Let $x_{1:P}$ denote a prompt or interaction prefix and $y_i = (y_{i,1}, \ldots, y_{i,R_i})$ the i-th of G responses sampled from it. Let $m_{i,t} \in \{0, 1\}$ select scored tokens and $N_i = \sum_t m_{i,t} > 0$. For trainable parameters θ and fixed rewards, advantages and response weights w_i, updates take the form

$$\mathcal{L}_{\mathrm{RL}}(\theta) = \sum_{i=1}^{G} w_i L_i(\theta), \qquad L_i(\theta) = \frac{1}{N_i} \sum_{t=1}^{R_i} m_{i,t} \ell(\rho_{i,t}(\theta), A_{i,t}, D_{i,t}(\theta)), \tag{1}$$

where $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x_{1:P}, y_{i,<t}) / \pi_{\mathrm{old}}(y_{i,t} \mid x_{1:P}, y_{i,<t})$, $A_{i,t}$ is an advantage, and $D_{i,t}$ optionally regularizes against a reference policy. Per-response averaging uses $w_i = 1/G$; global scored-token averaging uses $w_i = N_i / \sum_j N_j$. Proximal policy optimization (PPO) uses a clipped probability-ratio surrogate (Schulman et al., 2017), GRPO normalizes rewards within the response group (Shao et al., 2024) and DAPO modifies clipping and token aggregation (Yu et al., 2025); all fit Equation 1.

3.2 The Prefix as a Gradient Boundary

At optimizer step k, LongStraw keeps the prompt state

$$S_P^k = (Z_P^k, M_P^k), \qquad M_P^k = (\text{owner}, \text{position}, \text{boundary}, \text{version} = k), \tag{2}$$

where $Z_P^k = Z_P(\theta_k)$ holds the tensors each layer needs to continue from position P and M_P^k records their owner, positions, boundary and parameter version. The conditional objective at step k and its

gradient are

$$\widetilde{\mathcal{L}}_k(\theta) = \sum_i w_i L_i\big(\theta; \mathrm{stopgrad}(Z_P^k), M_P^k\big)\,, \qquad g_k = \nabla_\theta \widetilde{\mathcal{L}}_k(\theta)\Big|_{\theta=\theta_k}. \tag{3}$$

Full-sequence differentiation of $\mathcal{L}_{\mathrm{RL}} = \mathcal{L}(\theta, Z_P(\theta))$ also includes the prefix-construction gradient:

$$\nabla_\theta \mathcal{L}(\theta, Z_P(\theta)) = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{Z_P} + \frac{\partial \mathcal{L}}{\partial Z_P}\frac{\partial Z_P}{\partial \theta}. \tag{4}$$

At $\theta = \theta_k$, Equation 3 retains the first term, which trains adapter uses at tail and response positions, including the queries and recurrent reads over the captured prefix, and omits only the second, which would change how the prefix itself is encoded. Recapturing after every optimizer step keeps S_P^k current; older state would add drift that grows with each reused update (Section 5.2).

First answer token and differentiable prompt tails. The first answer distribution is read from the last prompt hidden state, $p(y_{i,1} \mid x_{1:P}) = \mathrm{softmax}(W_{\mathrm{out}}\,\mathrm{Norm}(h_P))_{y_{i,1}}$. With a tail of $b \geq 0$ prompt tokens, LongStraw captures $x_{1:P-b}$ without autograd and differentiates $x_{P-b+1:P} \| y_i$, scoring only answer tokens (Figure 2a); $b = 0$ is Equation 3, $b > 0$ replaces Z_P there with Z_{P-b}.

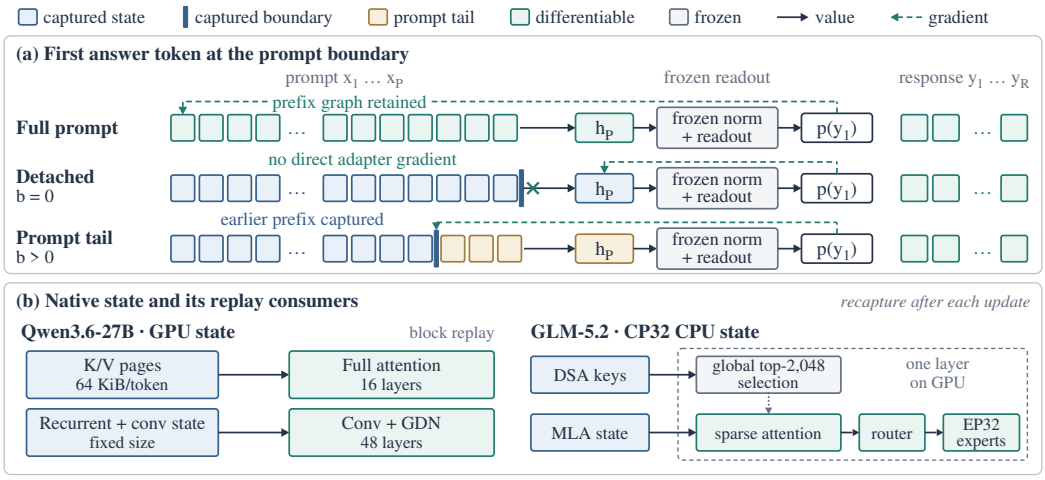


Figure 2: **The gradient boundary and the retained state.** (a) With frozen normalization and readout, detaching the last prompt state removes the first answer token's direct adapter gradient; a b-token tail restores it (Proposition 1). Solid/dashed arrows carry values/gradients. (b) Qwen K/V pages feed full attention; recurrent and convolution states feed Gated DeltaNet (GDN) and convolution replay. GLM stages CPU state sharded over 32 context-parallel ranks (CP32): DeepSeek Sparse Attention (DSA) keys select top-2,048 positions (dotted, no gradient); multi-head latent-attention (MLA) state feeds sparse attention. The routed expert path, 32-way expert-parallel (EP32), is shown; the shared expert and full architecture are in Appendix B.

Proposition 1 (First-token gradient)**.** *Let adapters θ act only inside decoder blocks, with embeddings,* Norm *and W_{out} frozen, as is standard for LoRA (Hu et al., 2022) and in all Qwen learning configurations, and let T_i be the $t = 1$ term of $w_i L_i$ in Equation 1. (a) If $b = 0$, then for any response weight, advantage and clipping rule,*

$$\nabla_\theta \log p(y_{i,1} \mid \mathrm{stopgrad}(h_P)) = 0 \quad \textit{and} \quad \nabla_\theta T_i = 0. \tag{5}$$

(b) If $b \geq 1$, $\nabla_\theta T_i$ equals the full-prompt gradient's contributions through adapter uses at positions $P - b + 1, \ldots, P$; those through positions $1, \ldots, P - b$ are omitted.

Detached updates still train later answer tokens and the end-of-sequence (EOS) token, and shared parameters can change $p(y_{i,1})$ indirectly. Yet the removed term carries weight $1/N_i$, or $1/2$ to $1/5$ for a one-to-four-token answer followed by EOS, and 71 of the 96 held-out 8K questions have such a reference; Section 5.1 measures the learning cost. Part (b) makes the repair local: any

$b \geq 1$ restores a direct path; we use $b = 32$ (Tail32), one replay block. Tail length alone sets the gradient boundary; capture blocks, state pages and replay blocks change memory and speed, not the objective. GLM-5.2 instead trains an output-head LoRA, giving the first answer token a direct gradient at $b = 0$. Appendix F gives the proof and a numerical check.

4 LONGSTRAW

LongStraw couples the prefix-state structure of Equation 2 with a capture–replay–recapture algorithm. Compact layer-specific tensors carry ownership, position, gradient-boundary and parameter-version metadata, separating shared prefixes from private extensions (Figure 2b). Capture builds this state without autograd; reverse replay differentiates one response block at a time; recapture refreshes it after updates (Algorithm 1). Paged K/V storage, FlashAttention-2 (FA2) and layer checkpointing provide the underlying operations for state storage and response recomputation.

4.1 WHAT MUST STAY ON THE GPU

Low-rank adapters and, on Qwen, 4-bit NormalFloat (NF4) base weights (Hu et al., 2022; Dettmers et al., 2023) reduce model storage to about 20 GiB, so sequence state and transient workspace set peak memory. For Qwen3.6-27B, only the 16 full-attention layers keep per-token state: BF16 K/V pages take 64 KiB per token, or 64 GiB at 1M. The 48 Gated DeltaNet (GDN) layers (Yang et al., 2025) keep 0.14 GiB of recurrent and convolution state at any length. The peak rises by 8.01 GiB from 128K to 256K, matching the 8 GiB page increment. At 1M, pages, recurrent state and weights account for 83.8 of the 98.81 GiB resource-run peak (Figure 4); the rest is transient workspace (Appendix Table 12). The 1M learning runs add temporary FA2 prefix dK/dV, peaking at 125.46 GiB (Table 2). Full-prompt differentiation with per-layer checkpointing would instead hold BF16 inputs to all 64 layers: 640 GiB at 1M, ten times the K/V pages. K/V pages, their gradients, recurrent state and weights alone need 147.8 GiB, above an H200's 140.4 GiB.

Views retain entire parent buffers. LongStraw copies captured state into compact pages, recording position, owner and role, and allocates persistent prefix dK/dV only when a gradient reaches it. Replaying one response at a time prevents its activations from accumulating across the G responses.

4.2 CAPTURE, REPLAY AND RECAPTURE

Algorithm 1 One LongStraw update at step k (Qwen instantiation). $w_i = 1/G$; L_i averages the unclipped surrogate $-\rho_{i,t}A_i$ over scored tokens; $\bar{r}_q$ and σ_q are the mean and standard deviation of rewards in group q; Clip_1 clips the global gradient norm to 1. Qwen 1M–4M runs use $Q = 8$, $G = 4$, $b = 32$, $P_0 = P - 256$, $n_{\mathrm{cap}} = 4{,}096$ and $n_{\mathrm{rep}} = 32$.

Input: policy θ_k; Q prompts $x^{(q)}_{1:P}$ sharing $x_{1:P_0}$; tail b; group size G.

1: $g \leftarrow 0$; capture the prompt state S^k (Equation 2) from $x_{1:P_0}$ under θ_k without autograd in n_{cap}-token blocks, copying each block's K/V to compact pages
2: **for** $q = 1, \ldots, Q$ **do**
3: extend S^k with $x^{(q)}_{P_0+1:P-b}$ without autograd
4: extend S^k with $x^{(q)}_{P-b+1:P}$ without autograd; sample $y_{1:G} \sim \pi_{\theta_k}(\cdot \mid x^{(q)}_{1:P})$, keeping sampled log-probabilities and rewards $r_{1:G}$; roll S^k back to $x^{(q)}_{1:P-b}$
5: $A_i \leftarrow (r_i - \bar{r}_q)/(Q\sigma_q)$; **if** $\sigma_q = 0$, skip to line 10
6: **for** $i = 1, \ldots, G$ **do**
7: run $x^{(q)}_{P-b+1:P} \| y_i$ from a private extension of $\mathrm{stopgrad}(S^k)$ without autograd in n_{rep}-token blocks, saving block-start state
8: **for** blocks j from last to first **do**
9: recompute block j with autograd; accumulate the block-j part of $\nabla_\theta(w_i L_i)$ into g; carry state and suffix dK/dV derivatives to block $j-1$; release block j
10: roll S^k back to $x_{1:P_0}$
11: **if** any $\sigma_q > 0$: $\theta_{k+1} \leftarrow \mathrm{AdamW}(\theta_k, \mathrm{Clip}_1(g))$, averaging g over CP ranks first; **else** $\theta_{k+1} \leftarrow \theta_k$
12: invalidate S^k; step $k+1$ recaptures under θ_{k+1}

Capture retains each layer's continuation state and frees attention, feed-forward, router and expert intermediates. The Q questions of a batch share the captured state of their first P_0 tokens; each extends it, rolling back to $P-b$ after sampling and to P_0 after discarding its private replay extensions. Gradients accumulate over responses, replayed one at a time, before one norm-clipped optimizer step. With parameters, advantages, masks and old log-probabilities fixed, serial replay yields the weighted sum of response gradients at the gradient boundary in exact arithmetic, including the $1/Q$ normalization of Algorithm 1 (Appendix D). An initial forward pass saves the state at each block boundary without retaining its computation graph. Backward replay starts at the last block. Each recomputed block receives state derivatives from later blocks and passes derivatives of its own starting state to the preceding block. Losses on later answer tokens can therefore propagate through earlier tail and response positions, with only one block's computation live at a time. Every block still uses the captured prompt state. Appendix B describes the states used by each architecture.

4.3 Model-Specific Placement

Where the state of Equation 2 lives depends on the architecture; CPn, EPn and TPn denote n-way context, expert and tensor parallelism. At 2M and 4M, Qwen keeps CP4- and CP8-sharded pages on GPU. Query-parallel FA2 gathers one K/V head at a time during backward to bound workspace (Appendix B). GLM-5.2 (GLM-5 Team, 2026; zai-org, 2026) keeps CP32-sharded multi-head latent-attention (MLA) state (DeepSeek-AI, 2024) and DeepSeek Sparse Attention (DSA) keys (DeepSeek-AI, 2025) on CPU, copying only owned pages and staging one layer for replay; capture bounds expert workspace with row chunks and output-buffer reuse (Appendix B.1). EP32 routes tokens to their expert owners (Lepikhin et al., 2021; Fedus et al., 2022; Liu et al., 2025a), and Megatron (Shoeybi et al., 2019) finalizes gradients after all G responses accumulate. vLLM rollout (Kwon et al., 2023) and Megatron training alternate with a shared policy and YaRN (Peng et al., 2024); cross-runtime policy ratios are approximate (Appendix A).

5 Evaluation

Qwen runs use Qwen3.6-27B (Qwen Team, 2026) with NF4 weights, BF16 compute, rank-16 FP32 LoRA on all attention, GDN and MLP projections and answer-F1 rewards; each rollout batch takes one on-policy AdamW step at 10^{-5}, so $\rho_{i,t}$ in the unclipped group-relative surrogate of Equation 1 is one in exact arithmetic. Learning runs use 32-token replay blocks and one, four and eight H200s at 1M, 2M and 4M, with static YaRN factors 4, 8 and 16; single-H200 resource runs use 4 and the 8K and 128K studies none. GLM uses rank-8 BF16 LoRA and factor-2 YaRN on 32 H200s. Appendix Table 6 summarizes setups; Appendix C has full protocols. Qwen memory is peak PyTorch allocation in GiB; token counts use $\mathrm{K} = 2^{10}$ and $\mathrm{M} = 2^{20}$, so 1M is 1,048,576 tokens.

5.1 Learning with a 32-Token Prompt Tail

At 8K, where full-prompt gradients still fit, a 32-token tail restores the learning that detachment largely loses (Table 1). The model answers SQuAD questions whose supporting passages are placed among other documents. Training and evaluation articles are disjoint. The three methods share prompts, initial adapters, forward computation and training order; only the prompt positions that receive gradients change. Native full differentiates the entire prompt, Tail32 its last 32 tokens, and detachment none. In every case, the loss scores answer tokens and EOS.

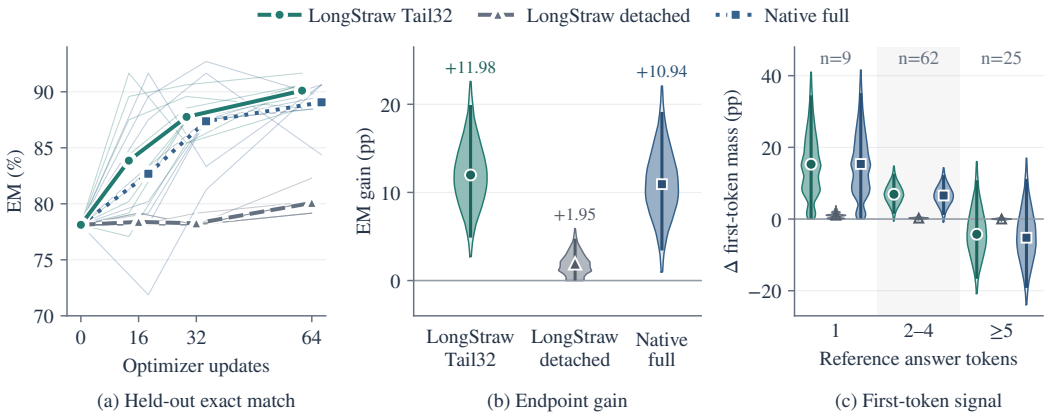
Native full differentiates a shared prefix graph (Liu et al., 2025b; Li et al., 2026a). EM rises 11.98 points with Tail32, 10.94 with Native full (Tail32 minus full: $+1.04$ $[-1.17, 3.37]$) and 1.95 with detachment, and Tail32 tracks Native full at every checkpoint (Figure 3a). Tail32 gains 10.03 points $[3.59, 17.30]$ over detachment for 2.74 GiB more peak memory.

Consistent with Proposition 1, the gain is largest where the first answer token carries the most loss weight: on one-token references, Native full and Tail32 EM each rise by 31.94 points and first-token mass by 15.37 and 15.29 points, versus 6.94 and 1.06 with detachment (Figure 3c; Appendix E).

In an exploratory 128K comparison, 13 seeds train for 12 updates with $R = 128$. Tail32 gains 1.68 EM points (95% interval $[0.24, 3.92]$), versus 1.76 with full-prompt gradients and 0 with detachment. Its gain differs from full-prompt training by -0.08 points $[-0.97, 0.74]$ (Appendix C.5).

Table 1: **Matched gradient scopes at 8K.** Eight seeds per arm, 64 updates, 96 held-out questions; all arms start at 78.13% EM and 90.67% F1. Gains are points; CIs use a paired seed/article bootstrap. Tail32 minus detached: EM +10.03 [3.59, 17.30]; Tail32 minus Native full: +1.04 [−1.17, 3.37].

	Differentiable	EM (%)			F1 (%)		
Gradient scope	**prompt tokens**	**Final**	**Gain**	**95% CI**	**Final**	**Gain**	**Peak (GiB)**
LongStraw Tail32	32	90.10	+11.98	[4.89, 19.92]	93.78	+3.12	45.10
LongStraw detached	0	80.08	+1.95	[0.00, 4.76]	91.23	+0.56	42.36
Native full	8,192	89.06	+10.94	[3.42, 19.13]	93.25	+2.58	75.85



(a) Held-out exact match (b) Endpoint gain (c) First-token signal

Figure 3: **A short prompt tail restores learning at 8K.** Eight seeds per arm, 96 held-out questions. (a) EM over training: thick lines are seed means and thin lines individual seeds. (b) EM gains after 64 updates. (c) Change in gold first-token probability mass by reference-answer length (n questions per stratum; descriptive). Shapes are kernel densities of the 20,000 paired seed/article bootstrap means (central 99%); darker cores span 95% intervals and marks are means; detached shapes in (c) sit near zero inside hollow marks. Appendix E gives F1 and the Tail32-minus-detached contrasts.

5.2 1M-Token Updates on One H200

LongStraw completes a 1M-token update on one H200 in 1,120.7 seconds at 98.81 GiB (Figure 4). The native baselines differentiate the same 32-token tail and responses under the same conditional objective, so differences reflect execution alone. Like LongStraw, they capture the detached prefix once without gradients in 4,096-token blocks, here into the Hugging Face K/V cache. Native tail keeps saved tensors on GPU; Native tail + CPU offloads them to host memory. All methods share inputs, initial adapters, $G = 4$ and fixed responses of $R = 512$ tokens. With this response length, full-prompt autograd runs out of memory at 64K on GPU and at 128K with CPU offload; the separate 128K learning study fits with $R = 128$ (Appendices C.2 and C.5).

Gradient-free prefix capture dominates the 1M update, taking 83% of its time and serving all G responses (Appendix Table 8). Native tail + CPU runs out of GPU memory during this capture, before any saved tensor exists to offload; LongStraw instead copies captured state into compact pages (Section 4.1). From 128K to 512K, LongStraw takes 20.4–33.6% less time than CPU offload and uses 14.2–41.2% less GPU memory. Host memory is 3.5–3.9 versus 43.7–147.2 GiB. Native tail is faster where it fits, but LongStraw uses 33.8% and 46.8% less GPU memory at 128K and 256K; Native tail runs out of memory at 512K (Appendix Table 7).

Storage and replay. Compact pages and lazy prefix gradients both lower peak memory. Each paired trial shares tokens and initial adapters and changes one storage choice. These tests use detached prompts and long responses on eight H200s. Across 20 paired trials from 32K to 1M, compact pages save 4.6–52.4% versus retained views, and lazy prefix dK/dV allocation saves 1.1–23.8% versus eager allocation, with identical gradients. Removing layer checkpointing causes all 20 trials to run out of memory (Appendix Figure 6a). In the separate single-H200 comparison, replaying the tail and each 512-token response as one block each cuts median 128K update time by a factor of 3.95 versus uniform 32-token replay (Appendix C.2).

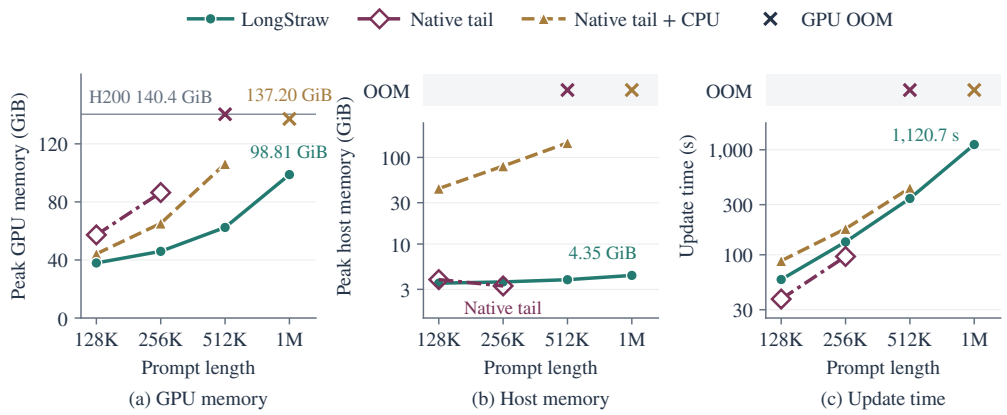


Figure 4: **LongStraw completes 1M-token updates on one H200.** (a) Peak allocated GPU memory, (b) peak host resident memory and (c) median update time after warm-up for $G = 4$ fixed 512-token responses, excluding generation. Native tail differentiates the same 32-token tail and responses with autograd; Native tail + CPU also offloads its saved tensors to host memory; LongStraw replays the tail and each response as one block each. Panels (b) and (c) use log scales; crosses mark GPU OOM, and Native tail is not run at 1M (Appendix C.2).

Recapture after updates. The prompt state is shared within an update and rebuilt after parameters change. Reusing it across updates would avoid the dominant capture cost, but the old state no longer represents the current policy. A fixed 1M diagnostic holds response tokens and updated parameters fixed and compares reused with recaptured state. Their mean absolute log-probability difference is 0.0137 after one update and 3.774 after eight (Appendix C.4).

Agreement with native autograd. Sharing one captured prompt state across the G responses is the common way to accelerate such updates (Liu et al., 2025b; Li et al., 2026a); combined with block replay, it changes the floating-point execution order. Against independent native autograd under the same conditional objective, LongStraw is exact at 32K on one rank, and distributed and uneven-page tests match all 992 adapter gradients and AdamW updates within 2×10^{-10} relative error. The numerical difference grows with context length. With deterministic kernels on both paths and the uniform 32-token replay of the learning runs, at 128K gradients differ by 0.024 relative L_2 (cosine 0.9997) and AdamW updates by 0.18, largely through first-step sign flips of near-zero coordinates. At 512K, the longest tested length at which native autograd still fits (with CPU offload), gradients differ by 0.058 (cosine 0.9984) and AdamW updates by 0.25 (Appendices C.2 and J).

5.3 Learning on Complete 1M Inputs

Tail32 improves EM on complete 1M inputs and gains more than detachment (Table 2). Its gain is $+2.03$ points (95% paired-seed interval $[1.09, 2.96]$). This exceeds the detached gain by 1.04 points $[0.21, 1.88]$. Leaving out each archive keeps gains positive ($+0.31$ to $+4.78$ points).

Each input appends a SQuAD question (Rajpurkar et al., 2016) to a complete million-token archive. Its supporting paragraph lies between 10% and 90% of the archive, among whole paragraphs from other articles. The evidence is thus far before the differentiable tail, but remains available to every response. Training and evaluation use disjoint article pools. Each seed trains on 128 questions from 64 articles for four epochs. The 96 evaluation questions come from 48 held-out articles and share four archives. Across 18 seeds, Tail32 and detachment receive the same archives and paired initial adapters. Greedy evaluation reads the complete inputs after checkpoint reload (Appendices H and I).

Question-only EM declines under Tail32: it ends at 19.33%, 1.50 points below the initial model (Appendix Table 15). Training sharpens the first answer token, consistent with Proposition 1: gold first-token mass under Tail32 rises from 73.28% to 80.68% (Appendix I).

Table 2: **Complete-input QA learning with NF4 Qwen3.6-27B.** All rows score 96 paired held-out questions; gains are points. Train is wall-clock hours with generation; peak is allocated GiB per GPU, at 1M with FA2 prefix K/V-derivative temporaries, unlike Figure 4 (Appendices C and G).

Input	LongStraw	EM (%)	ΔEM	95% CI	ΔF1	H200s	Train (h)	Peak (GiB)
1M on one H200: 18-seed means; 64 one-archive 8-question batches, 32-token generation; paired-seed CIs								
1M	Tail32	76.04 → 78.07	+2.03	[1.09, 2.96]	+0.73	1	20.46	125.46
	detached	76.04 → 77.03	+0.98	[0.47, 1.50]	+0.08	1	19.98	125.18
2M on four H200s (CP4), 4M on eight (CP8): 16 updates on 128 training questions, 512-token generation								
2M	Tail32	56.25 → 57.29	+1.04		+2.27	4	9.78	77.74
4M	Tail32	27.08 → 33.33	+6.25		+4.88	8	27.58	95.75

5.4 Scaling to 2M and 4M Tokens

Paired initial and final evaluations in single-seed 2M and 4M runs show EM and F1 gains (Table 2). F1 rises in every archive at 2M; at 4M, EM rises in every archive, with seven questions improving and one declining. Doubling GPUs at 4M keeps retained state at 51.8 GiB per rank, but growing global workspace raises peak memory from 77.74 to 95.75 GiB (Appendix Table 12 and Figure 11).

5.5 Extension to GLM-5.2 at 2M Tokens

We extend LongStraw to the mixture-of-experts GLM-5.2 (GLM-5 Team, 2026; zai-org, 2026), completing a 2M-token update at 107.8 GiB per rank on 32 NVIDIA H20 GPUs. This feasibility test extends one DAPO math prompt. A conventional Megatron update retaining the prompt graph completes at 32K but runs out of memory at 2M in all three TP/CP/EP layouts tested (Appendix C.1).

On 32 H200s, LongStraw completes five online policy updates across the first ten question groups, each a 2,097,152-token input that embeds a HotpotQA question (Yang et al., 2018) and its context passages in Wikipedia background text; groups whose four answer-F1 rewards tie skip the optimizer. After each applied update, the next group's responses are sampled from the updated policy. Sampling four responses per group takes a median of 38.24 minutes, and device memory peaks at 110.4–129.2 GiB per GPU. Updates detach the prompt ($b = 0$) and train rank-8 LoRA adapters that include the output head, which gives the first answer token a direct gradient (Section 3.2; Appendix A). An output-head log-probability check stopped the run before held-out evaluation (Appendix A).

5.6 Discussion

Context length and gradient scope serve different purposes. Our QA gains persist when supporting evidence lies outside the differentiated region, and a short tail retains much of full-prompt learning's gain at 8K. This suggests budgeting gradients separately for using and constructing representations.

Moving tensors to CPU need not reduce GPU memory. In the immediate FA2 backward probe, original K/V remain live, so restoring offloaded tensors creates extra device copies. At 1M, retaining these tensors instead lowers both memory and backward time (Appendix Table 13). Effective placement depends on ownership and release points.

Bounding backward shifts the bottleneck toward prefix construction. Sharing state amortizes this cost, but reuse after updates introduces growing score drift (Section 5.2). Current-policy caches need both shared text and a parameter version, making state lifetime part of the update algorithm.

6 Conclusion

LongStraw combines a compact, versioned prefix-state structure with a capture–replay–recapture algorithm to train on complete long histories under memory constraints. Qwen learns from million-token QA inputs, and GLM executes distributed online updates. The framework separates access to historical evidence from the extent of prompt differentiation. This opens a broader direction: understanding how architecture, training data and answer length shape useful gradient scope, and adapting that scope to the available memory budget.

AI USE STATEMENT

AI tools assisted with literature research, language polishing, and code debugging.

REPRODUCIBILITY STATEMENT

Section 4 and Appendices B, D and G specify the method and its implementation; Section 5 and Appendices A and C give the evaluation protocols; Appendix H describes the million-token QA dataset; Appendices E, I and J give the learning protocols and numerical validation; Appendix K describes positional settings. The anonymous supplement includes core code and selected protocols.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395, 2024. doi: 10.18653/v1/2024.findings-emnlp.74. URL https://aclanthology.org/2024.findings-emnlp.74/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. IndexCache: Accelerating sparse attention via cross-layer index reuse. *arXiv preprint arXiv:2603.12201*, 2026.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. URL https://arxiv.org/abs/1604.06174.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.12307.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, 2019. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285/.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. URL https://arxiv.org/abs/2512.02556.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Hao Ge, Junda Feng, Qi Huang, Fangcheng Fu, Xiaonan Nie, Lei Zuo, Haibin Lin, Bin Cui, and Xin Liu. ByteScale: Efficient scaling of LLM training with a 2048k context length on more than 12,000 GPUs. *arXiv preprint arXiv:2502.21231*, 2025. URL https://arxiv.org/abs/2502.21231.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, et al. DeepSpeed Ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023. URL https://arxiv.org/abs/2309.14509.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692v2. Version 2.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Dacheng Li, Rulin Shao, Anze Xie, Eric P. Xing, Xuezhe Ma, Ion Stoica, Joseph E. Gonzalez, and Hao Zhang. DistFlashAttn: Distributed memory-efficient attention for long-context LLMs training. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=pUEDkZyPDl.

Pengbo Li, Feiyuan Zhang, Guangming Sheng, Guangxin He, Di Chai, Ziniu Li, Taiqiang Wu, Wenyu Mao, Binhang Yuan, and Kai Chen. Schedule-level shared-prefix reuse for LLM RL training. *arXiv preprint arXiv:2606.01143*, 2026a. URL https://arxiv.org/abs/2606.01143.

Wenhao Li, Yuxin Zhang, Gen Luo, Daohai Yu, and Rongrong Ji. Training long-context LLMs efficiently via chunk-wise optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2691–2700, 2025. doi: 10.18653/v1/2025.findings-acl.138. URL https://aclanthology.org/2025.findings-acl.138/.

Wenhao Li, Daohai Yu, Gen Luo, Yuxin Zhang, Fei Chao, Rongrong Ji, Yifan Wu, Jiaxin Liu, Ziyang Gong, and Zimu Liao. Out of the Memory Barrier: A highly memory efficient training system for LLMs with million-token contexts. *arXiv preprint arXiv:2602.02108*, 2026b. URL https://arxiv.org/abs/2602.02108.

Dennis Liu, Zijie Yan, Xin Yao, Tong Liu, Vijay Korthikanti, Evan Wu, et al. MoE parallel folding: Heterogeneous parallelism mappings for efficient large-scale MoE model training with Megatron Core. *arXiv preprint arXiv:2504.14960*, 2025a.

Hao Liu, Matei Zaharia, and Pieter Abbeel. RingAttention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, 2024.

Zikang Liu, Tongtian Yue, Yepeng Tang, Longteng Guo, Junxian Cai, Qingbin Liu, Xi Chen, and Jing Liu. Prefix Grouper: Efficient GRPO training through shared-prefix forward. *arXiv preprint arXiv:2506.05433*, 2025b. URL https://arxiv.org/abs/2506.05433.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.00071.

Qwen Team. Qwen3.6-27B configuration. Hugging Face model configuration, 2026. URL https://huggingface.co/Qwen/Qwen3.6-27B. Pinned revision 6a9e13bd6fc8f0983b9b99948120bc37f49c13e9; accessed July 16, 2026.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pp. 2383–2392, 2016. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. URL https://arxiv.org/abs/1909.08053.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. QwenLong-L1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025. URL https://arxiv.org/abs/2505.17667.

Siyuan Wang, Gaokai Zhang, Li Lyna Zhang, Ning Shang, Fan Yang, Dongyao Chen, and Mao Yang. LoongRL: Reinforcement learning for advanced reasoning over long contexts. *arXiv preprint arXiv:2510.19363*, 2025. URL https://arxiv.org/abs/2510.19363.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990. doi: 10.1162/neco.1990.2.4.490.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813.

Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, Bo Li, and Bryan Catanzaro. From 128K to 4M: Efficient training of ultra-long context large language models. *arXiv preprint arXiv:2504.06214*, 2025. URL https://arxiv.org/abs/2504.06214.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Jinghan Yao, Sam Ade Jacobs, Masahiro Tanaka, Olatunji Ruwase, Hari Subramoni, and Dhabaleswar K. Panda. Training ultra long context language model with fully pipelined distributed transformer. In *Proceedings of Machine Learning and Systems*, 2025. URL https://arxiv.org/abs/2408.16978.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.02259.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

zai-org. GLM-5.2 model card and configuration. Hugging Face model card and configuration, 2026. URL https://huggingface.co/zai-org/GLM-5.2. Pinned revision b4734de4facf877f85769a911abafc5283eab3d9; accessed August 3, 2026.

A GLM-5.2 Updates at Two Million Tokens

This appendix details the GLM-5.2 run of Section 5.5, in which LongStraw completes five policy updates across ten 2M-token question groups on 32 H200s. The reported window covers the first ten groups, which have NVML memory logs; updates are applied at groups 1, 2, 7, 8 and 10. Table 3 reports each group.

Table 3: **GLM-5.2 sampling and policy updates at 2M on 32 H200s.** Each row is one question group with four sequential responses. Sampling time is the wall-clock time until all four responses return. Applied counts one optimizer step across all 32 ranks, and cumulative counts updates through that group. These are consecutive measurements, not independent timing repetitions.

Group	Four response lengths (tokens)	Sampling (min)	Applied	Cumulative
1	3, 28, 3, 4	38.38	1	1
2	6, 8, 9, 367	39.95	1	2
3	4, 3, 164, 3	38.58	0	2
4	7, 302, 6, 7	39.39	0	2
5	7, 6, 8, 7	37.84	0	2
6	6, 6, 3, 6	37.74	0	2
7	7, 5, 512, 5	40.78	1	3
8	4, 41, 6, 4	38.11	1	4
9	5, 5, 5, 7	37.79	0	4
10	5, 3, 4, 3	37.71	1	5
Total		386.27	5	5

Inputs and update configuration. Each input contains exactly 2,097,152 tokens under the official GLM chat template. The constructed HotpotQA (Yang et al., 2018) inputs preserve the question and all ten context paragraphs within natural Wikipedia background; training draws from 16 fixed questions. The model samples four responses per group, sequentially, with temperature 1, top-$p = 0.95$ and a 512-token cap with natural EOS. Training uses rank-8 LoRA, learning rate 10^{-6} and answer F1 as the reward. The prompt is detached ($b = 0$), and a trained LoRA module on the output head gives the first answer token a direct gradient through the readout; Proposition 1 assumes a frozen readout and applies to the Qwen configurations. The clipped surrogate uses $\epsilon = 0.2$, no reference KL penalty ($\beta = 0$) and global scored-token averaging, with approximate policy ratios from raw log-probabilities of the two runtimes. Groups with zero reward variance skip the optimizer

without redrawing. Sampling uses eight-way tensor parallelism with four pipeline stages (TP8/PP4), and training uses TP1/CP32/EP32; both use the same GPUs in turn under factor-2 YaRN.

Sampling and device memory. The ten groups produce 40 responses of 3–512 tokens. Median four-response sampling time is 38.24 minutes, with a range of 37.71–40.78. Sampling totals 6.44 hours; each group is timed until all four responses return, excluding capture, replay and optimization. Within every group, the first response has no prefix-cache hit and the following three each reuse 2,097,088 tokens. The first two applied updates take 28.26 and 28.90 minutes from capture through replay and result collection, excluding checkpoint export. Periodic NVML measurements give per-device peaks of 110.42–129.15 GiB (Table 4). Sampling and training never overlap in time.

Table 4: **Sampled device-memory peaks during the GLM 2M run.** Each GPU contributes its maximum NVIDIA Management Library (NVML) memory reading over the observation window; each row gives the minimum and maximum of those eight device peaks. Each node has 1,169 valid measurements, approximately 30.31 seconds apart. Periodic sampling gives lower bounds on true peaks; these are device-memory measurements, not PyTorch allocated-memory values.

Node	H200s	Lowest device peak (GiB)	Highest device peak (GiB)
1	8	110.42	110.94
2	8	128.45	128.97
3	8	128.45	128.97
4	8	128.65	129.15

Each applied update in the ten-group window is followed by checkpoint export and sampling from the updated policy; every sampler worker is checked to load the updated adapters, including a nonzero output-head module. Each update also checks that log-softmax over the complete 154,880-way output-head logits reproduces the recorded first-token log-probabilities within 2×10^{-5}. The planned 16-group run stopped during group 12, when this check failed after passing at every earlier update, before held-out evaluation. By then it had started 12 groups, taken seven optimizer steps and exported the first six updates.

Capture and response lifetimes. At 2M, each CP32 rank processes 65,536 prompt tokens. Top-8 routing creates 524,288 token–expert assignments; a balanced BF16 dispatched-input matrix at hidden width 6,144 occupies 6 GiB per rank before router metadata and other intermediates. Expert-load imbalance can increase this configuration-derived size. Capture runs without autograd and releases expert intermediates after use, while replay differentiates only the responses. CPU-resident latent-attention state and sparse-selection keys are staged one layer at a time; Appendix B.1 details the capture allocation strategy and its distinct response-backward path.

Compact diagnostic output. The implementation records parameter hashes and gradient norms, finite-value counts and nonzero counts instead of serializing full gradient tensors. For one applied 2M update, the before/after records across all ranks total 26.0 MiB. Computing these summaries still transfers parameters and gradients to CPU; compact output reduces serialization volume but does not remove the associated transfer or synchronization cost.

B Architecture-Specific State and Replay

Qwen state and distributed attention. Qwen has 64 layers, each with a dense gated feed-forward network; 48 use GDN and 16 use full attention. It keeps recurrent and convolution states and compact K/V pages on GPU. CP8 assigns persistent K/V pages to their owning ranks. The distributed attention implementation merges local attention through global log-sum-exp normalization and combines response $dQ/dK/dV$ before the preceding layer. The formal 2M/4M executions instead use query-parallel FlashAttention-2 (FA2): forward gathers K/V and distributes query blocks, while backward gathers one K/V head at a time to bound temporary workspace. FA2 computes temporary prefix derivatives but retains only the suffix contributions; no prefix derivative buffer is stored between calls. Separate tail and response blocks reduce packing overhead.

GLM state and expert placement. GLM combines 3 dense and 75 MoE layers. It keeps CP32-sharded multi-head latent-attention (MLA) state (DeepSeek-AI, 2024) from all 78 layers on CPU. DeepSeek Sparse Attention (DSA) index keys (DeepSeek-AI, 2025) come from 21 producer layers; the other 57 IndexShare layers reuse their producer's selection, following the model configuration (zai-org, 2026) and cross-layer index reuse (Bai et al., 2026). A 2M prompt has 1,024 64-token pages per rank; replay stages one layer at a time. Each producer merges local DSA candidates into a global top 2,048 before value composition, and its IndexShare consumers reuse these indices. The discrete selection receives no gradient. EP32 routes each token to its top-8 of 256 experts; one shared expert bypasses routing and adds to the combined output. Replay reconstructs selection at producer layers and recomputes routing, expert inputs and combine buffers, then releases layer workspace. Megatron finalizes gradients after all G responses, before the optimizer step.

Rollout and training layouts. vLLM rollout and Megatron training time-share the 32 GPUs in the layouts of Appendix A; training captures its own state under the same policy and YaRN configuration. Parameter transfer preserves active adapter contributions in both runtimes. Appendix C.1 describes the output-head restriction in the H20 experiment. Figure 5 summarizes the retained state, reconstructed workspaces, their release points and backward paths of both architectures.

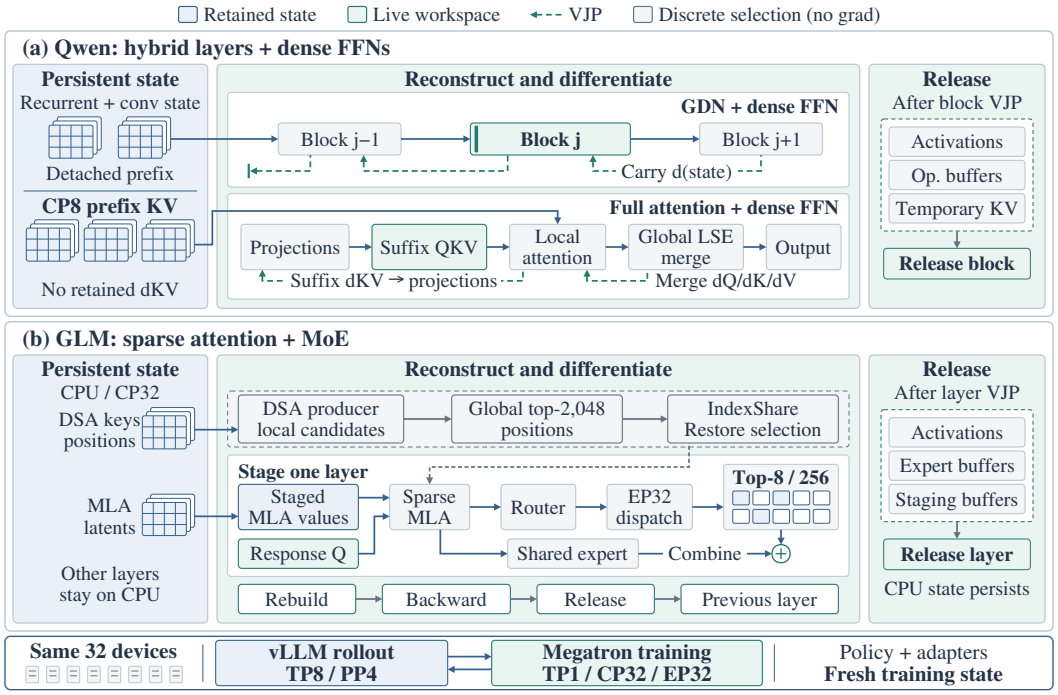


Figure 5: **Architecture-specific state and replay.** (a) Qwen reuses recurrent boundaries and sharded K/V pages, reconstructs suffix activations, and carries derivatives across blocks. (b) GLM stages one layer of CPU state, restores discrete IndexShare selections, and reconstructs attention and expert computation. Backward releases consumed workspaces while retaining detached prefix state.

B.1 GLM Memory Controls Across Capture and Replay

The GLM implementation in Section 4.3 controls both persistent attention state and transient expert workspace. Table 5 summarizes the configured bounds and where each mechanism applies. The mechanisms below describe the implementation; their individual contributions to end-to-end time and peak memory were not isolated by matched ablations.

Copy only the owned state. Capture copies each CP-owned MLA or DSA page directly into its final CPU allocation, avoiding a full-component copy followed by page clones. In one applied 2M update, all 2,496 rank–layer capture records show CPU-resident pages and zero full-component

Table 5: **GLM controls persistent state and temporary workspace at different phases.** The bounds below specify the 2M implementation; conditional paths are named explicitly.

Phase / path	Memory / I/O cost	Control or configured bound
Capture	MLA / DSA pages	Copy only owned pages to CPU; stage one layer at a time back on the GPU for replay.
Capture	Expert outputs	Evaluate 2,048-row MLP chunks; outputs reuse consumed input rows when shapes and dtypes agree.
Capture, non-fused	Token permutation	Reorder contiguous CUDA inputs in place using a 4-column stripe and a 32-bit row-index map.
Capture	Routing, shared expert	Release dispatch references after use; shared-expert output can wait on CPU until combine.
Conventional DSA	Selection indices	Release after the last consumer; keep on pinned CPU when recomputation needs them.
Response backward	Saved activations	One checkpoint per decoder layer; use the original sequential MLP and sort operators.
Update diagnostics	Gradient tensors	Save parameter hashes and gradient summaries; full gradient transfers to CPU remain.

copy bytes, with nonzero owned-page copies. Replay transfers only the current layer's owned pages back to the device, preserving global page indices for causal attention.

Bound temporary expert allocations. During no-gradient capture, sequential experts evaluate bounded row chunks and write each result into the input rows already consumed, avoiding a separate output list and its concatenation. This reuse is enabled when input and output shapes and dtypes agree. The non-fused token sort can reorder its receive matrix in place, using a narrow column stripe and a row-index map. For the balanced 2M dispatch geometry in Appendix A, the receive matrix occupies 6 GiB; a four-column BF16 scratch stripe occupies 4 MiB, with a further 2 MiB for 32-bit row indices. These are allocation sizes derived from the geometry: the receive matrix and other live tensors remain allocated. The capture path also drops routing and dispatch references after their consumers and can temporarily move the shared-expert output to CPU until combination.

Match lifetimes to recomputation. DSA producer layers share selection indices with later IndexShare consumers. The conventional DSA path releases each shared holder after its final forward consumer. When layer checkpointing requires the holder again, that path instead stores it in pinned CPU memory, reloads it for recomputation and releases it after its final backward use. CPU-resident response replay uses a separate sparse-attention helper rather than this holder-offload path. Response replay enters an autograd-safe mode: the sequential expert MLP and token sorting use the original operators, and capture-time buffer reuse, shared-expert offload and nested MoE checkpointing are disabled. LoRA updates merge into fresh buffers. With CPU-resident prefix state, replay still checkpoints each decoder layer and re-enters the same safe mode during backward recomputation.

Additional chunking paths. The framework also provides 65,536-row expert-LoRA chunks and 8,192-token output-projection and vocabulary cross-entropy chunks. These limits do not split work in the reported response replay: the adapter projects response rows directly, and the response cap is 512 tokens. During capture, the smaller sequential-expert chunks already bound the input to expert LoRA. These general implementation controls are distinct from the decoder's capture and replay blocks and do not establish further memory savings for the response sizes used here.

C Evaluation Protocols and Extended Results

GPU peaks report PyTorch allocated memory unless labeled otherwise; values labeled GiB use 2^{30} B and those labeled GB use 10^9 B. The 1M, 2M and 4M learning studies replay responses in 32-token blocks, with generation capped at 32 tokens for 1M and 512 for 2M/4M; a 512-token answer can therefore span sixteen response blocks. Optimized single-GPU resource profiles instead use a 32-token tail block, a 512-token response block, 32-token pages and 4,096-token capture blocks. Timing repeats measure execution cost, not learning variance. Table 6 lists each study's setup.

Table 6: **Experimental setups.** R is the generation cap in learning runs and the response length otherwise. Slashes in b list arms as Tail32/detached/native full, where $b = P$ keeps full-prompt gradients. Block is the response replay block; 32+512 pairs a 32-token tail block with a 512-token response block, and whole replays each response at once. YaRN gives the static factor; none keeps the pretrained 262,144-position range. Updates are counted per run or storage case; resource runs count warm timed updates per length.

Study	GPUs	P	R	G	b	Block	YaRN	Seeds	Updates	Where
Qwen3.6-27B: NF4 weights, BF16 compute, rank-16 FP32 LoRA										
8K QA	1 H200	8K	32	4	32/0/P	32	none	2101–2108	64	5.1, E
Resources	1 H200	128K–1M	512	4	32	32+512	4	–	3–20	5.2, C.2
Storage	8 H200, CP8	32K–256K, 1M	8,192	2	0	2,048	none	201–220	1	5.2, C.3
128K QA	1 H200	128K	128	4	0/P	128	none	311–323	12	C.5
1M QA	1 H200	1M	32	4	32/0	32	4	4101–4118	50–64	5.3, I
2M/4M QA	4 H200, CP4 8 H200, CP8	2M 4M	512	4	32	32	8 16	4101	16	5.4, I.1
GLM-5.2: rank-8 BF16 LoRA; training at TP1/CP32/EP32, sampling at TP8/PP4										
2M update	32 H20	2M	5	2	0	whole	2	–	1	5.5, C.1
2M online	32 H200	2M	512	4	0	whole	2	–	5	5.5, A

Qwen LoRA ($\alpha = 32$, no dropout) adapts `q/k/v/o_proj` in the 16 full-attention layers, `in_proj_qkv/z/b/a` and `out_proj` in the 48 GDN layers and `gate/up/down_proj` in all 64 MLPs (496 modules, 992 adapter tensors); embeddings, norms and the output head stay frozen. Each Qwen rollout batch takes one on-policy AdamW step. The 8K and 128K studies keep the pretrained rotary range of 262,144 positions. Static YaRN with original maximum position 262,144 uses factor 4 for single-GPU resource profiles and 1M learning (configured maxima 1,050,624 and 1,048,608) and factors 8 and 16 at 2M and 4M (2,097,664 and 4,194,816); GLM uses factor-2 YaRN.

Clipped actor loss. The GLM experiments instantiate ℓ in Equation 1 as a clipped surrogate; with ratio-clip half-width ϵ and KL weight β, the per-response loss is

$$L_i(\theta) = \frac{1}{N_i} \sum_{t=1}^{R_i} m_{i,t} \left[-\min(\rho_{i,t} A_{i,t}, \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) A_{i,t}) + \beta D_{i,t} \right]. \tag{6}$$

C.1 GLM Feasibility and Parity Protocols

Full-sequence feasibility. The conventional baseline and LongStraw use the same GLM-5.2 checkpoint, rank-8 BF16 LoRA, and 32 H20 GPUs. The Megatron baseline retains the full prompt graph and completes an optimizer update at 32K. At 2M, TP2/CP16/EP32, TP1/CP32/EP32, and TP4/CP8/EP32 all exhaust GPU memory during forward or backward. In TP4/CP8/EP32, a 2.5 GiB allocation fails with 62 MiB free and 136.2 GiB allocated by PyTorch.

LongStraw runs at TP1/CP32/EP32 with detached-prefix gradients, an exact 2,097,152-token prompt, two distinct five-token completions, four scored tokens per completion, and a DAPO rule-based reward. Backward passes for both responses and gradient accumulation complete on every rank, with global DSA selection for each rank and response, finite globally finalized gradients, and one optimizer step. Peak allocated memory is 107.8 GiB per rank.

The selected prompt prepends 2,097,059 tokens of reference context to an intact 93-token target chat. The context contains 11,855 other DAPO source items with role-tagged problems and verified reference answers, excluding candidate target rows; its body occupies 2,097,006 tokens, with a 32-token header and 21-token footer. The target question appears at the end and is answerable from that question alone, so the prompt length measures execution extent, not use of distant information.

We selected the target from earlier samples: up to eight source questions with eight samples each, requiring distinct equal-length completions with mixed rewards. Two questions were tried; the selected pair was “Answer: 2” and “Answer: 4”, each five tokens with rewards -1 and $+1$. The

experiment characterizes one update on selected policy samples. It does not estimate unbiased rollout quality or evaluate a multi-step interaction trajectory.

Same-state 32K parity. The conventional and response-only arms load the same base weights and LoRA state and consume the same tokens, selected positions, old/reference scores, advantages, and objective parameters. The ratio-one case tests the first-update neighborhood; the active-clipping case forces one lower- and one upper-clipped token. We compare selected log probabilities, scalar loss, parameter names and shapes, zero/nonzero masks, and every gradient shard. The relative gradient L_2 errors of the two cases, 1.171% and 1.163%, exceed the prespecified 1% threshold. A 2% tolerance was selected after observing them and is reported only descriptively.

Sampler-to-trainer transfer. Online RL must also fit response generation within the GPU budget. Keeping separate rollout and training models resident would add their memory requirements even when their computation is sequential. The GLM execution instead time-shares the same 32 H20s between a vLLM sampler at TP8/PP4 and Megatron training at TP1/CP32/EP32. The training runtime releases its devices before the sampler starts. Response tokens, old-policy log-probabilities, and rewards remain valid after the sampler releases its devices; the training runtime then restores the policy and captures its own model-native prompt state.

The transfer preserves the policy snapshot and the position configuration. Both runtimes use identical checkpoint parameters and the same explicitly configured factor-2 YaRN extension for the 2M prompt. In this single-update experiment, the sampler omits the output-head LoRA module, whose contribution is zero at transfer; nonzero output-head updates would require exporting that module. Time-sharing avoids holding both models at once, at the cost of releasing and restoring runtimes; that time counts toward end-to-end online execution.

C.2 Single-GPU Native Resource Comparisons

We compare native tail-gradient autograd (Native tail), native tail with CPU saved-tensor offload (Native tail + CPU) and LongStraw on one H200. All use Qwen3.6-27B NF4 weights, BF16 computation, rank-16 FP32 LoRA, $b = 32$, $G = 4$, and $R = 512$. Both native arms capture the detached prefix once without gradients in 4,096-token blocks, reuse it for all four responses and back-propagate each 512-token response as it is scored; LongStraw instead stores compact pages and replays the tail and responses from them. Within each length, supplied token sequences and initial adapters are identical. Static YaRN uses factor 4 at every length, with maximum position 1,050,624 ($2^{20} + 2{,}048$), which covers a 1M prompt and a 512-token response. Each observation starts from the same adapter weights and empty AdamW state. The native controls have three warm observations per length. Twenty warm LongStraw updates at 128K come from four separate single-H200 processes, each performing five paired repetitions of uniform and separate blocks. The 256K, 512K and 1M profiles each use one cold and three warm updates; we report full observed ranges.

Table 7 reports complete-update allocated peaks, host resident set size (RSS) and elapsed time. Timings include prompt capture, response scoring and backward, and optimization; model loading, initialization resets, file I/O and response generation are excluded. Each process receives a separate GPU. Cold updates are excluded from the timing summaries, and warm GPU peaks are the maximum across warm repetitions. CPU saved-tensor offload has a 512 GiB process budget. The native measurements and optimized profiles use matching inputs and initial adapters but were collected in separate sessions; their differences are descriptive, not paired speedups. The 128K comparison of uniform and separate blocks is paired on the same GPU within each process.

Separate tail and response blocks. Uniform blocking reconstructs 17 blocks per response: one 32-token tail and sixteen 32-token response blocks. Separate blocking reconstructs two blocks while retaining the same tail boundary and head-by-head FA2 execution. Across four responses and 16 attention layers, attention VJP calls decrease from 1,088 to 128 per update, and head-level VJPs from 4,352 to 512. Each larger block attends to the same causal prefix positions. Flash Linear Attention (FLA) kernels and checkpointing are unchanged. Across the 20 warm pairs at 128K, median update time falls from 229.78 to 58.15 s ($3.95\times$) and maximum warm allocation from 40.04 to 37.93 GiB; the mean paired time reduction is 75.00% (95% interval $[74.87, 75.11]$).

Table 7: **Native and LongStraw resources on one H200.** NF4 Qwen3.6-27B, rank-16 FP32 LoRA, $G = 4$, $R = 512$, $b = 32$. GPU cold and warm columns give maximum observed allocated peaks in GiB; host is maximum warm process RSS in GiB. LongStraw has 20 warm observations at 128K and three elsewhere; native controls have three. LongStraw replays separate 32-token tail and 512-token response blocks. Median and range (minimum–maximum) give warm update time in seconds. Each repeat resets adapters and AdamW; these are timing repetitions, not training seeds. Native tail autograd runs out of GPU memory (OOM) at 512K and is not run at 1M.

Prompt	Implementation	GPU cold	GPU warm	Host	Median (s)	Range (s)
128K	Native tail	56.17	57.27	3.89	37.9	37.7–38.1
128K	Native tail + CPU	44.93	44.22	43.74	87.6	85.1–87.6
128K	LongStraw	37.93	37.93	3.53	58.1	57.0–66.9
256K	Native tail	85.35	86.43	3.28	96.3	96.2–96.6
256K	Native tail + CPU	65.90	65.21	79.41	177.0	175.1–177.4
256K	LongStraw	45.94	45.94	3.66	132.6	131.8–133.1
512K	Native tail	OOM	OOM	–	–	–
512K	Native tail + CPU	106.93	106.23	147.19	431.8	429.2–435.1
512K	LongStraw	62.39	62.42	3.88	343.7	343.0–344.0
1M	Native tail + CPU	OOM	OOM	–	–	–
1M	LongStraw	98.81	98.81	4.35	1120.7	1113.0–1123.2

Figure 6b and Table 8 decompose the warm update nearest each arm's median total (229.45 and 58.19 s) into synchronized, disjoint phases that sum to that update's measured total. At 128K, capture takes 34.21 s with uniform blocks and 33.93 s with separate blocks; response scoring and backward take 195.08 and 24.20 s. The reduction is concentrated in response execution. Head tiling, FA2 and the gradient boundary remain fixed, so this comparison measures the combined effect of response block size on reconstruction, packing and kernel calls.

Table 8: **Complete-update phases with uniform and separate replay blocks.** Seconds for actual warm observations nearest each arm's median total. The 128K block comparison has 20 paired repetitions across four single-GPU processes; longer contexts use three descriptive repetitions with separate blocks. Uniform replay uses 32-token blocks; separate replay uses a 32-token tail and 512-token response blocks. Replay includes reconstruction and backward. Other includes optimization and residual time; phases add to the observed total before rounding.

Prompt	Schedule	Capture	Replay	Other	Total
128K	Uniform blocks	34.21	195.08	0.16	229.45
128K	Separate blocks	33.93	24.20	0.06	58.19
256K	Separate blocks	91.17	41.37	0.08	132.62
512K	Separate blocks	271.65	72.03	0.06	343.74
1M	Separate blocks	935.05	185.59	0.07	1,120.72

The 128K latency interval resamples five paired observations within each of the four fixed process/GPU strata, using 10,000 bootstrap samples and equal stratum weights. It describes timing variation conditional on the measured hardware and fixed input. The shorter three-repeat profiles are descriptive and do not supply confidence intervals.

The GPU-resident Native tail implementation fails at 512K and is not run at 1M. Native tail + CPU fails during gradient-free prefix capture at 1M, with 137.20 GiB allocated by PyTorch and 51.16 GiB of host RSS; saved-tensor offload moves only tensors saved for backward. Full-prompt gradients are a separate objective; we test both 4,096-token prefix capture with 512-token query tiles and single-prefix capture with query extent $P - 32$. In both profiles, GPU-resident full-gradient execution fails during backward at 64K. At 128K the chunked profile fails during capture and the single-prefix profile during response scoring; CPU saved-tensor offload fails during backward in both profiles. Chunked and single-prefix offload use 276.48 and 147.47 GiB of host RSS, below the host budget. All are CUDA memory failures; longer lengths are not run.

An independent 128K comparison of native GPU autograd, which differentiates each 512-token response in one pass, and LongStraw replay in uniform 32-token blocks, both with FP32 recurrent state, fixed FLA launches and deterministic FA2 backward, yields finite gradients for all 992 tensors: gradient relative L_2 difference 0.02399 and cosine 0.999712; AdamW-update relative L_2 difference 0.18263 and cosine 0.983322. The maximum absolute selected-token log-probability difference is 0.06871. Both runs start from empty AdamW state, so the first step is approximately $-\mathrm{lr} \cdot \mathrm{sign}(g)$: near-zero coordinates can flip sign between the two gradients, and the maximum absolute update difference is 2.0×10^{-5}, twice the learning rate of 10^{-5}. Because $\sqrt{2(1 - 0.983322)} = 0.18263$ reproduces the relative difference, the two updates have nearly equal norms, and the first AdamW step amplifies the 0.024 gradient difference to 0.18. Replay and native autograd are therefore not numerically identical at 128K. At 256K, gradient and AdamW-update differences are 0.02386 (cosine 0.999715) and 0.18314. GPU-resident native autograd fails at 512K, so the 512K reference offloads to CPU; at 128K and 256K, offloaded and GPU-resident native gradients and updates are bit-identical. At 512K, the differences are 0.05789 (cosine 0.998368) and 0.25052. At 1M, native autograd with CPU offload runs out of memory, whereas LongStraw completes the update.

C.3 Qwen Storage and Checkpointing Ablations

Figure 6a and Table 9 isolate storage choices under a fixed response workload. These tests use Qwen3.6-27B NF4 rank-16 QLoRA, CP8 on eight H200s, $b = 0$, $R = 8{,}192$, $G = 2$, 2,048-token response blocks, 4,096-token prefix blocks, 64-token pages, and whole-layer response checkpointing. Response MLP work uses 512-token micro-blocks; prefix MLP work uses 2,048. Each case is one synthetic unclipped response-conditional update, not a QA training run.

The component matrix uses 20 independent paired blocks, four each at 32K, 64K, 128K, 256K, and 1M, with disjoint seeds 201–220 and the same response geometry. Within each block, four separately initialized arms share identical token inputs and initial adapters. Each alternative changes one storage or checkpointing choice. The baseline runs first; the other three arms rotate order across execution shards. We compare the complete 992-tensor gradient against the paired baseline, with prespecified cosine at least 0.99, relative L_2 at most 0.05, and loss difference at most 0.005. Both retained views and eager prefix gradient storage have cosine 1 and relative error 0 in every completed pair. Removing checkpointing causes explicit CUDA allocation failures in all 20 cases. Each paired block is one independent observation, so the 80 configurations are not 80 independent repetitions.

Table 9: **One-component Qwen ablations on eight H200s.** Four independent paired seeds at each length give 20 blocks and 80 configurations, with $R = 8{,}192$, $G = 2$. Retaining views (Views) and eagerly allocating prefix dK/dV (Eager) each reproduce all 992 baseline gradients exactly in every completed pair. Peaks and increases over LongStraw (Δ) are allocated GiB per rank; differences use unrounded measurements. Removing layer checkpointing (No ckpt.) causes CUDA OOM in all 20 cases.

Prompt	LongStraw Peak	Views Peak	Views Δ	Eager Peak	Eager Δ	No ckpt. Outcome
32K	43.47	45.55	+2.08	43.97	+0.50	OOM (4/4)
64K	43.72	47.55	+3.83	44.72	+1.00	OOM (4/4)
128K	44.22	51.55	+7.33	46.22	+2.00	OOM (4/4)
256K	45.23	59.56	+14.33	49.23	+4.00	OOM (4/4)
1M	51.25	107.58	+56.33	67.25	+16.00	OOM (4/4)
Complete	20/20	20/20	20/20	20/20	20/20	0/20

C.4 State Recapture after Parameter Updates

Parameter updates change the prefix computation, so reusing its old state can change response scores even when the response tokens are fixed. A fixed 1M diagnostic compares recaptured and reused state under the same updated parameters. After 1, 2, 4, and 8 updates, mean absolute log-probability differences are 0.0137, 0.0250, 3.136, and 3.774; signed loss differences are 0.124%, 0.038%, −22.81%, and −9.33%. All learning studies therefore recapture the prefix at every update.

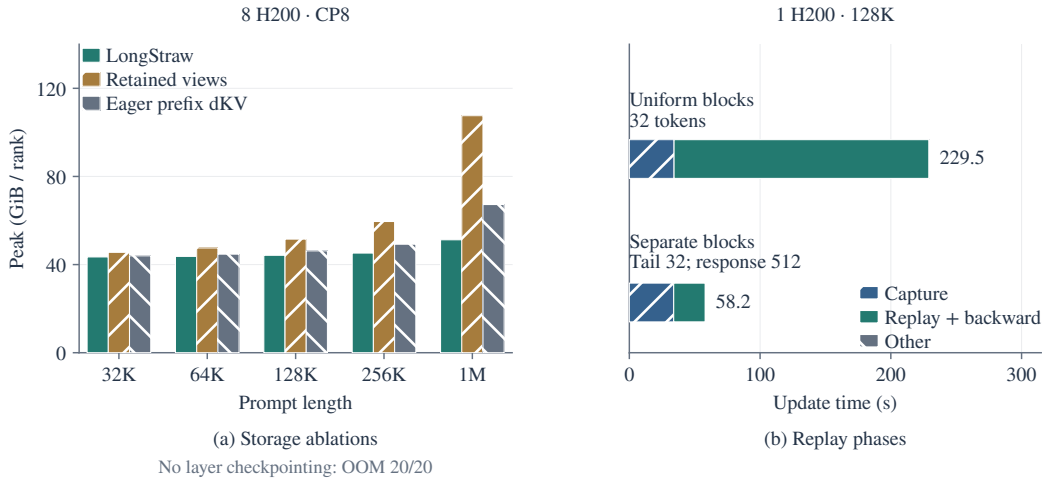


Figure 6: **Storage and replay interventions in distinct Qwen configurations.** (a) Eight H200s (CP8), $b = 0$, $G = 2$, $R = 8{,}192$: median maximum-rank allocated GiB from four paired storage trials per length. Retained views and eager prefix dK/dV reproduce baseline gradients; removing layer checkpointing causes OOM in all 20 cases. (b) One H200 at 128K, FA2, $b = 32$, $G = 4$, $R = 512$: uniform 32-token blocks versus separate 32-token tail and 512-token response blocks. Bars decompose updates selected independently nearest each arm's median from 20 warm pairs in four fixed process/GPU strata. Capture, replay-plus-backward and other time sum to the selected total, not a sum of phase medians. Tables 9 and 8 give the values.

C.5 128K QA across Prompt Gradient Scopes

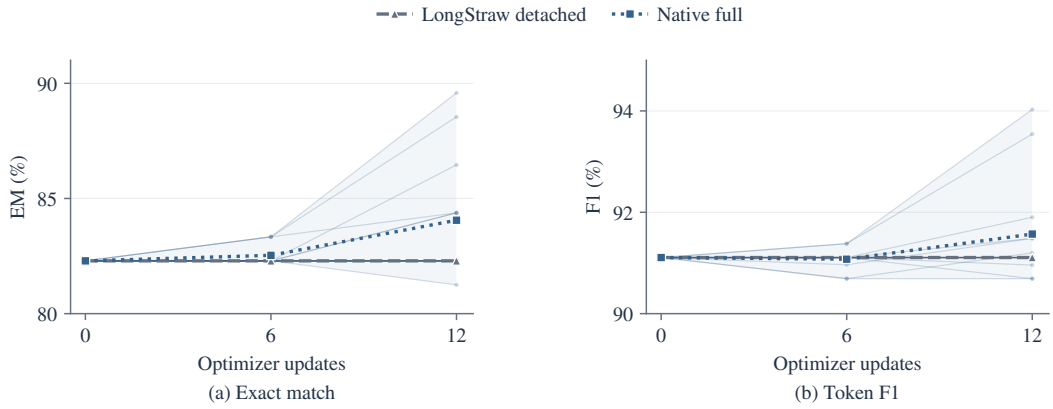


Figure 7: **Full-prompt and detached QA scores over 12 fresh-prefix updates.** Thirteen seeds per method; 96 fixed questions at 128K. Strong lines/markers show means, faint marks show all seeds; detached LongStraw curves coincide. Shading spans the observed seed range, not a confidence interval. Segments connect measured checkpoints without smoothing.

We construct 131,072-token contexts by placing an original SQuAD question's complete evidence passage among other documents (Rajpurkar et al., 2016). Training, development, and evaluation article pools are disjoint. The fixed evaluation set contains 96 questions from 40 articles; its scores are not official SQuAD results. In the original detached–native comparison, thirteen independent seeds initialize LoRA and sampling, with matched initial adapters and prompt order within each pair. Each method uses Qwen3.6-27B NF4 rank-16 QLoRA with pretrained rotary positions on one H200, $R = 128$, $G = 4$, sequential generation, and AdamW at 10^{-5}. Answer F1 supplies group-normalized advantages for an unclipped on-policy loss without KL, including the first response token and EOS. The native baseline uses layer checkpointing and CPU saved-tensor staging while

retaining the full prompt VJP; detached LongStraw recaptures the prompt before every update. Thus paired runs share prompts, seeds and evaluation questions but have different backward objectives.

Each run trains on 12 groups with nonzero reward variance, drawing at most 128 groups within 12 hours; skipped groups are retained. Evaluations occur at steps 0, 6, and 12; endpoint conditions are intact context, equal-length removal of the target passage, and question only. Adapter, optimizer, and RNG states are checkpointed at all three steps and reloaded for evaluation. Repeated initial evaluations produce identical predictions. Matched 8K/128K forward comparisons before and after training have zero maximum log-probability difference under common fixed kernel configurations. All 26 original runs complete training, with 992 finite named gradients and changed parameters at every update. Table 10 reports exact-match changes for intact inputs and both information controls.

Table 10: **Constructed 128K QA: matched changes across information controls.** Seed-mean EM (%) before and after 12 updates; gains and 95% intervals are percentage points. All 13 seeds contribute. Only detached intact EM is primary; other conditions and contrasts are exploratory.

		LongStraw detached		Native full	
Context	**Initial**	**Final**	**Gain [95% CI]**	**Final**	**Gain [95% CI]**
Intact	82.29	82.29	$0.00^{\dagger}$	84.05	$+1.76\ [0.17, 4.10]$
Target removed	5.21	4.65	$-0.56\ [-2.08, 0.00]$	5.61	$+0.40\ [-0.32, 1.79]$
Question only	17.71	18.43	$+0.72\ [-0.33, 2.74]$	17.71	$0.00\ [-1.86, 1.71]$

†Every observed primary change is zero, producing an empirical $[0, 0]$ interval; the prespecified analysis treats this as no observed change, not a bound on unseen effects or evidence of population equivalence.

The primary estimator averages paired question-level EM changes within each seed, then weights seeds equally. Its percentile interval uses 20,000 paired bootstrap draws over independent seed and article axes. The prespecified positive rule requires a mean gain of at least 10 points and a lower bound above zero; the detached arm does not meet it.

Intact-context F1 is 91.11% at both detached endpoints and rises from 91.11% to 91.57% for the native baseline, whose exploratory change interval is $[-0.35, +1.56]$ points. The detached-minus-native EM change difference is -1.76 points with exploratory interval $[-4.10, -0.17]$; secondary intervals are not adjusted for multiple comparisons. Removing the target evidence reduces initial EM from 82.29% to 5.21%, showing dependence on that evidence in this constructed task.

Short-tail follow-up. Tail32 uses the same 13 seeds, initial adapters, prompt order, 12-update schedule and evaluation questions. It differentiates the last 32 prompt tokens and responses, with 32-token pages and replay blocks. All 13 runs complete training and evaluation. This exploratory comparison reuses the original seed/article bootstrap estimator and the two completed baselines.

Table 11: **Tail32 gains EM at 128K while detached scores remain unchanged.** Means over 13 seeds after 12 updates on the same 96 questions. All methods start at 82.29% EM and 91.11% F1. Gains and 95% paired seed/article intervals are percentage points; comparisons are exploratory.

Gradient scope	**EM (%)**	**EM gain [95% CI]**	**F1 (%)**	**F1 gain [95% CI]**
Tail32	83.97	$+1.68\ [0.24, 3.92]$	91.45	$+0.35\ [-0.65, 1.46]$
Native full	84.05	$+1.76\ [0.17, 4.10]$	91.57	$+0.46\ [-0.35, 1.56]$
Detached	82.29	$0.00^{\dagger}$	91.11	$0.00^{\dagger}$

†All observed changes are zero; the same interpretation as Table 10 applies.

Tail32 raises EM on six seeds and leaves it unchanged on seven (Table 11); F1 rises on four, is unchanged on three and falls on six. Its EM gain exceeds detached by 1.68 points $[0.24, 3.92]$. The Tail32-minus-native gain differences are $-0.08\ [-0.97, 0.74]$ for EM and $-0.12\ [-0.61, 0.30]$ for F1. These unadjusted intervals describe paired differences; they do not establish equivalence.

The initial adapters and intact-input per-question EM/F1 match the original runs. On five seeds, two initial predictions differ in wording without changing those scores; some information-control predictions and scores also differ. The comparison therefore uses each run's own initial checkpoint and does not assume bitwise reproduction across the original and follow-up runs.

D Distributed Attention and Gradient Accumulation

This appendix gives the identities behind sharded attention and serial replay. The storage ablations use dense-sharded response attention. Persistent prefix-gradient pages are omitted, while response K/V derivatives are reduced to their owning ranks before adapter gradients are synchronized.

Distributed attention and accumulation identities. The sharded backend composes local maxima m_r, normalizers ℓ_r, and unnormalized value sums u_r through

$$m = \max_r m_r, \qquad \ell = \sum_r e^{m_r - m}\ell_r, \qquad o = \frac{\sum_r e^{m_r - m}u_r}{\ell}. \tag{7}$$

For fixed parameters, masks, weights, advantages, and old/reference scores,

$$\sum_{q=1}^{Q}\sum_{i=1}^{G} w_i \partial_\theta L_{q,i}(\theta; S_q^k) = \partial_\theta \left[\sum_{q=1}^{Q}\sum_{i=1}^{G} w_i L_{q,i}(\theta; S_q^k)\right], \tag{8}$$

where S_q^k is the captured state extended to question q's boundary and the $1/Q$ normalization is carried by the advantages A_i (Algorithm 1). Within one response, reverse block accumulation applies the chain rule block by block: each recomputed block receives the recurrent-state, convolution-state and suffix K/V cotangents carried from later blocks, so its contributions sum to the response gradient. This identity holds in ideal arithmetic; Appendix J checks numerical agreement.

E Matched 8K Gradient-Boundary Learning Study

The matched 8K study compares full-prompt gradients, LongStraw detached and Tail32 with eight seeds and 64 effective updates per arm. Tail32 reduces peak allocation from 75.85 to 45.10 GiB (40.5%) and reaches 90.10% EM, compared with 89.06% for full-prompt gradients (Table 1). Its training gain is 11.98 points $[4.89, 19.92]$; the paired gain over complete detachment is 10.03 points $[3.59, 17.30]$. Figures 8 and 9 report learning curves and effects by reference-answer length.

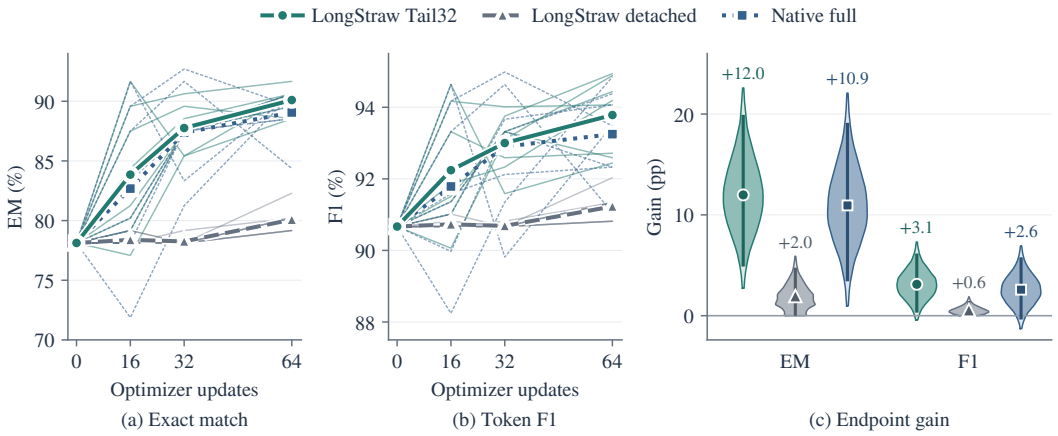


Figure 8: **The complete 8K learning curves.** Eight seeds, 96 fixed questions. Strong lines/markers show means; thin lines show individual seeds, which share the same step-0 scores (78.13% EM, 90.67% F1). In (c), shapes are Gaussian kernel densities of the 20,000 paired seed/article bootstrap mean gains (central 99%), scaled to a common maximum width; darker cores and bars span 95% intervals, and printed numbers give the means. Per-seed gains are the 64-update values of the thin lines minus this shared start; Figure 9a plots the per-seed EM gains. Straight segments join measured checkpoints. Figure 7 reports results for the separate 128K study.

Native full, LongStraw detached and Tail32 (a 32-token differentiable tail) use seeds 2101–2108. Each run completes 64 effective updates, with a maximum of 512 online groups, $P = 8{,}192$, $R = 32$, $G = 4$, temperature 1, the full vocabulary, and no forced minimum response length.

All arms use the same native prefix/tail/response forward phases, NF4 base weights, FP32 rank-16 LoRA, FP32 recurrence, deterministic FA2, fixed FLA launch settings and pretrained rotary positions without YaRN. The Native full arm shares one live prefix graph and differentiates the sum of response objectives once. AdamW uses learning rate 10^{-5}, betas $(0.9, 0.999)$, epsilon 10^{-8}, no weight decay, and gradient clipping at norm 1 after group accumulation. The objective remains the unclipped response surrogate with actual sampled old log probabilities. First-update comparisons with independent native autograd satisfy the numerical tolerances for every method seed under the same objective (Appendix J). The arms thus differ only in gradient scope.

The SQuAD construction of Appendix C.5 supplies 256 training questions and 96 evaluation questions from 40 articles disjoint from training. These evaluation questions also appear in the 128K study, so the two evaluations are not independent. Development used six separate questions. Training order is shuffled by seed and shared between arms. Evaluation uses checkpoints 0, 16, 32, and 64; checkpoints 16/32/64 are saved and reloaded before generation.

The prespecified primary rule requires a final tail-arm mean EM gain of at least two percentage points and a positive lower endpoint of a two-sided 95% interval, with all eight seeds completing training and the numerical checks. Two points is roughly two additional correct answers per 96 questions. The endpoint is fixed at 64 updates, with intermediate evaluations at 16 and 32. We use 20,000 paired crossed seed/article bootstrap draws, sampling eight seeds and 40 articles with replacement and retaining all questions in sampled articles. All 96 questions receive equal weight.

EM changes are $+10.94$ points $[3.42, 19.13]$ for Native full, $+1.95$ $[0.00, 4.76]$ for LongStraw detached, and $+11.98$ $[4.89, 19.92]$ for LongStraw Tail32. The tail meets the prespecified primary criterion. Its paired excess gain over detachment is 10.03 points $[3.59, 17.30]$, and its paired difference from full gradients is $+1.04$ points $[-1.17, 3.37]$ in EM and $+0.53$ $[-0.32, 1.58]$ in F1; these secondary comparisons do not establish superiority over full gradients. Initial F1 is 90.67%; final F1 is 93.25%, 91.23%, and 93.78%, respectively. The tail F1 change is $+3.12$ points $[0.31, 6.16]$.

The number of online groups needed for 64 updates ranges from 242 to 383 for full gradients, 241–275 for detachment, and 270–385 for the tail. The remaining 178–319, 177–211, and 206–321 groups have zero reward variance and produce no optimizer update. Complete run times include setup, sampling, diagnostics, training, and evaluation: 5,213–6,414 s, 4,332–4,950 s, and 5,591–6,625 s. Their different sampled responses prevent a matched speed comparison.

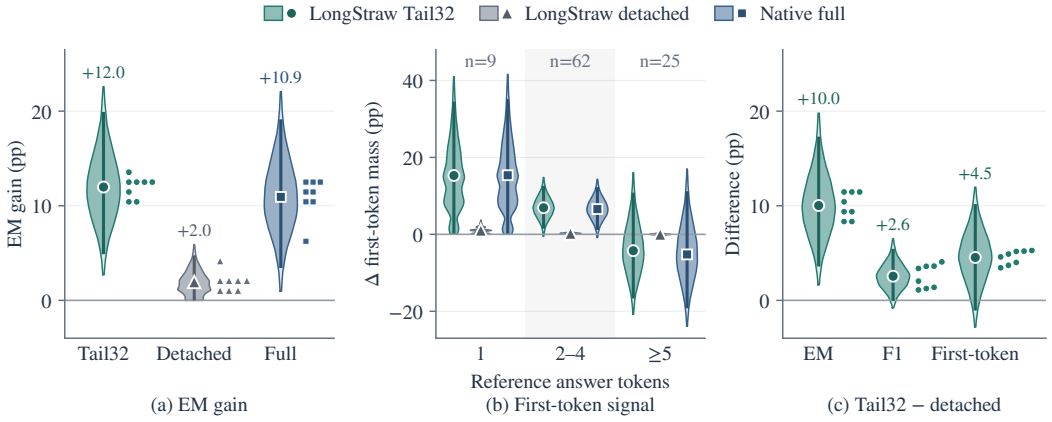


Figure 9: **Boundary and answer-length diagnostics.** (a) EM gains. (b) First-token probability-mass changes by minimum reference-answer token count (n questions per stratum), including leading-space variants and deduplicating token IDs. (c) Tail32-minus-detached contrasts. Shapes are Gaussian kernel densities of the 20,000 paired seed/article bootstrap means (central 99%), scaled to a common maximum width within each panel; darker cores and bars span 95% intervals, and large marks are means, printed in (a,c). Small marks beside the shapes in (a,c) are per-seed values, set side by side when closer than one mark width. In (b), a light band separates strata; the detached arm's changes stay near zero, so its shapes are nearly flat and largely hidden behind its markers. Strata are descriptive.

There are 9 one-token, 62 two-to-four-token, and 25 longer reference answers. For one-token references, full and tail EM each increase by 31.94 points, versus 6.94 for detachment, with wide intervals. First-token mass increases by 15.37, 1.06, and 15.29 points for full, detached, and tail arms. In the two-to-four-token stratum, full, detached, and tail EM changes are 9.88, 0.40, and 11.29 points; first-token mass changes are 6.57, 0.24, and 6.91 points. Tail intervals are $[4.30, 19.17]$ for EM and $[1.48, 12.62]$ for first-token mass in this stratum. For longer references, tail EM changes by 6.50 points $[-8.52, 20.24]$ and first-token mass by -4.25 $[-16.66, 10.82]$; these data are inconclusive. The first-token mass proxy does not cover every normalized correct answer. Across all questions, the tail mass change is 4.79 points $[-0.90, 10.53]$.

Length-matched answer-document removal yields initial EM 0% and final EM 5.21%, 0%, and 5.21% for full, detached, and tail arms. Question-only EM starts at 17.71% and ends at 17.71%, 17.71%, and 18.10%; the tail change is 0.39 points $[-1.81, 2.52]$. The large gap to intact-context performance supports the relevance of the supplied documents.

Sampling and scoring arithmetic. Native incremental generation and batched training scoring differ in finite precision, even at the same parameter snapshot. We retain the actual sampled log probabilities in the importance-ratio denominator rather than setting ratios to one. Across 512 effective detached-arm updates, the median per-group maximum selected-log-probability difference is 0.0797, the worst is 0.8738, and scored-token mean ratios range from 0.9664 to 1.0333. Corresponding tail values are 0.0735, 0.4740, and 0.9746–1.0215. The log-ratio clamp at ± 20 is inactive. All arms use the same sampler and matched training operators.

F Proof of Proposition 1

Proof. (a) The adapters reach $p_\theta(\cdot \mid x_{1:P})$ only through h_P, Norm and W_{out}. At $b = 0$, h_P is a stopgrad constant and the other two are frozen, so the first-token distribution is constant in θ; hence $\rho_{i,1}$, $D_{i,1}$ and ℓ, including any clipping, are constant as well. (b) Treat the use of θ at each position j as an untied copy $\theta^{(j)}$ and identify the copies after differentiation. By causality, the captured state Z_{P-b} depends only on $\theta^{(j)}$ with $j \le P - b$, so stopgrad removes exactly the contributions of those copies. Recomputing positions after $P - b$ from the same value of Z_{P-b} reproduces the forward values and the partial derivatives with respect to the remaining copies. The weight $1/N_i$ follows from Equation 1 with $m_{i,1} = 1$; all statements hold in exact arithmetic. □

Numerical check. A four-layer random Qwen hybrid with rank-4 LoRA and frozen base weights and readout tests Proposition 1 across five initializations, using two one-token answers with rewards $[-1, +1]$. At $b = 0$, adapter gradients and one-step answer log-odds changes are zero. At $b = 1$, gradient norms are 0.286–0.543 and rewarded-answer relative log probability increases by 0.000165–0.000587 after SGD at 10^{-3}. All $b = 4$ and full-prompt $b = 8$ cases also yield nonzero gradients and positive changes. This isolates the boundary mechanism; the matched 8K study (Section 5.1) and Appendix C.5 measure its learning effect.

G Memory Placement and Checkpointing

Analytic GPU state. Table 12 derives the retained state of NF4 Qwen3.6-27B from its configuration: 64 layers of hidden size 5,120, of which 16 are full-attention layers with four K/V heads of dimension 256 and 48 are GDN layers with 48 value heads of 128×128 FP32 recurrent state. The 32-token tail is recomputed, so $P - 32$ prompt tokens are paged. Only K/V pages grow with length. The measured peak exceeds the analytic sum by 15.0 GiB at 1M and by 25.9 and 43.9 GiB per rank at 2M and 4M; this remainder is transient workspace rather than retained state. Per-rank retained state is identical at 2M and 4M, yet the 4M peak is 18.0 GiB higher, so part of that workspace grows with global prompt length rather than with local state.

Immediate attention-backward staging. In the CP1 official-FA2 reverse sweep, saved-tensor placement affects the lifetime of temporary attention buffers. The outer CPU saved-tensor hook also applies to the differentiable FA2 call whose VJP is immediately consumed inside the custom backward. Its complete K/V tensors stay alive until the calling function returns; copying the inner

Table 12: **Analytic GPU state for NF4 Qwen3.6-27B.** The 1M column holds one prompt on one H200; the 2M/CP4 and 4M/CP8 runs both hold 524,288 prompt tokens per rank. LongStraw retains the first three rows. The last two rows give what full-prompt gradients with per-layer checkpoints would hold. Weights count the output head in FP32 (17.3 GiB in total if BF16). 1 KiB $= 1{,}024$ B.

State	Precision and layout	Per token	1M, one GPU	Per CP rank
K/V pages	BF16; $16 \times 4 \times 256$, K and V	64 KiB	64.0 GiB	32.0 GiB
GDN state	FP32 recurrent, BF16 convolution	–	0.14 GiB	0.14 GiB
Weights and adapters	NF4 linears, BF16 embedding	–	19.7 GiB	19.7 GiB
Analytic sum			83.8 GiB	51.8 GiB
Measured peak	allocated		98.81 GiB	77.74 / 95.75 GiB
Layer-input checkpoints	BF16; $64 \times 5{,}120$	640 KiB	640 GiB	320 GiB
Prefix dK/dV	BF16; K/V shape	64 KiB	64 GiB	32 GiB

saved tensors to CPU cannot release those originals, and restoring them creates additional GPU storage. We retain the inner saved tensors only until that immediate VJP completes, preserving outer query staging and surrounding layer checkpoints without changing the numerical operations.

The control uses one H200, the same NF4 base weights, rank-16 FP32 adapters, FP32 recurrent state, a 256-token question segment with a differentiable 32-token tail, and YaRN factor 4 with original maximum position 262,144. Two fixed questions share a natural-text archive. For each question, the supplied responses are its reference phrase and `Unknown`, each followed by EOS within a 32-token response buffer, with advantages $[-0.5, +0.5]$. The response groups are supplied and fixed across both modes. Both modes use the same captured archive, initial parameters, and empty AdamW state. Each receives a complete warmup before four measured runs ordered CPU, retained, retained, CPU. Timing and peak allocation cover the backward interval; capture, scoring, clipping, optimizer execution, and serialization are outside that interval. Table 13 compares backward time and peak allocation under the two saved-tensor placements.

Table 13: **Immediate saved-tensor retention: backward time and memory tradeoff.** Times are medians of two measured runs after warmup; the ratio divides the CPU-staged median by the retained median. Peaks are allocated GPU GiB. These supplied-response comparisons measure the backward stage only.

	Backward time (s)			Peak memory (GiB)		
Prompt	CPU staged	Retained	Ratio	CPU staged	Retained	Saved
32K	47.84	36.37	1.32×	29.03	29.03	0.00
1M	440.02	67.57	6.51×	128.58	124.58	4.00

At 32K, an independent native-autograd reference gives relative gradient and AdamW-delta errors 6.43×10^{-9} and 2.26×10^{-7}, with zero pre/post selected-score difference. Within each length, all four measured runs have bitwise-identical values for all 992 gradients, parameter deltas, updated parameters, pre-step selected scores, and loss. Both lengths use the same implementation, with 128 attention backwards per measured phase. At 1M, retention cuts the backward median from 440.02 to 67.57 s and peak allocation by 4.0007 GiB; both include full-prefix K/V derivatives.

Native checkpoint references. For the native baseline, the checkpoint wrappers store only metadata snapshots of attention caches. An 8K comparison against uncheckpointed autograd reproduces all 992 gradients and updates, the loss, sampling scores, and post-update scores exactly. Removing cache-array references from the checkpoint metadata reduces peak allocation from 71.46 to 69.91 GiB; the uncheckpointed disk-staged reference uses 28.04 GiB. Retaining cache metadata across eight updates preserves adapter/optimizer values and the 75.85-GiB peak.

Outer saved-tensor placement. With immediate inner retention enabled, we separately compare CPU and device storage for outer query-continuation intermediates. Removing this outer transfer preserves layer checkpointing and full-prefix derivative computation. Both modes receive a full warmup. Measurements then follow CPU, device, device, CPU order, each with the same initial

parameter state and the same supplied responses. Table 14 isolates the placement of outer saved tensors while keeping the inner saved tensors identical.

Table 14: **Outer saved-tensor placement with inner retention enabled.** Times are medians of two synchronized backward measurements after warmup; the ratio is the CPU median divided by the GPU median. Memory change is GPU minus CPU placement; comparisons fix prompt length.

	Backward time (s)			Peak memory (GiB)		
Prompt	**CPU saves**	**GPU saves**	**Ratio**	**CPU saves**	**GPU saves**	**Change**
32K	32.09	13.14	2.44×	29.03	31.45	+2.42
1M	63.21	40.28	1.57×	124.58	124.60	+0.02

Both placements yield bitwise equality of all 992 gradients, parameter deltas, updated parameters, complete AdamW states, loss and pre-update selected scores across placements and repeats. At 32K, a separately computed native-autograd reference also agrees with the gradients and post-update scores. The 1M comparison uses the same implementation. GPU saving adds 2.42 GiB at 32K but only 20 MiB at 1M. The long-input learning experiments use retained inner tensors and GPU storage for outer saved tensors. Native-autograd comparisons on the question-answering dataset cover both the differentiable-tail and detached execution modes at 32K.

H Million-Token Question-Answering Dataset

Fixed data construction. Disjoint pools of SQuAD articles (Rajpurkar et al., 2016) provide 128 training questions from 64 articles, 12 development questions from 12, and 96 evaluation questions from 48. Their questions share 16, four and four archives, respectively. Each prompt combines a 1,048,320-token archive with an exact 256-token question segment. Supporting paragraphs lie between 10% and 90% of the archive, so evidence ends 105,060–943,681 tokens before the prompt end. Distractors are whole SQuAD v1.1 paragraphs from other articles: the pool contains 14,921 paragraphs and 2,507,772 tokens, excluding all target articles, and each archive draws from it in its own seeded order. YaRN factor 4 (original maximum position 262,144) extends positions to 1,048,608 including the response, beyond Qwen3.6-27B's advertised 1,010,000-token context.

Annotation procedure. All 236 original questions were checked against source questions, articles, support text, answer offsets, token arrays and archive positions. Model-assisted semantic review with separate adjudication identified 15 clear inconsistencies: 11 question texts were corrected and incorrect answer alternatives removed in four cases. Another 23 ambiguous cases retain their annotations. IDs, splits, source articles, distractors and archive tokens remain fixed; both annotation versions and source offsets are retained. Review preceded main evaluation predictions, although some development outputs had been examined; it was neither expert-led nor fully blinded.

Information controls. The last-8K control keeps the final 8,192 original prompt tokens and resets positions to zero using the same arm's position configuration. Question-only inputs omit documents.

I Million-Token Learning Results

Protocol. The two full-input arms span seeds 4101–4118, totaling 36 runs, using the evaluation and archives of Appendix H. Each seed shuffles 16 training archives for four epochs: 64 training batches of eight fixed questions, each with $G = 4$ responses, totaling 512 groups and 2,048 responses. Sampling uses temperature 1 and a 32-token limit, without vocabulary truncation or minimum response length. Zero-variance groups stay in the batch denominator; zero gradients skip optimization without redrawing or extending training. Both arms use NF4 Qwen3.6-27B, BF16 computation, rank-16 FP32 LoRA, FP32 recurrence and one H200. AdamW uses learning rate 10^{-5}, betas $(0.9, 0.999)$, epsilon 10^{-8}, no weight decay and norm-1 clipping. Initial adapters are paired by seed. Tail32 differentiates the last 32 prompt tokens and responses; LongStraw detached differentiates responses only. Both use 1M inputs and static YaRN4, retaining inner saved tensors and storing outer saved tensors on GPU. Greedy evaluation after checkpoint reload runs on intact inputs at batches 0, 32 and 64 and on the information controls at batches 0 and 64. Each run is limited

to 72 hours. The prespecified primary criterion requires mean tail EM gain of at least five points and a positive lower two-sided 95% paired-seed t bound. The observed $+2.03$ points $[1.09, 2.96]$ (raw $p = 0.000261$) falls short. Under the prespecified order, the tail-minus-detached gain is descriptive. Seed intervals condition on these fixed questions and archives and require nonzero variance.

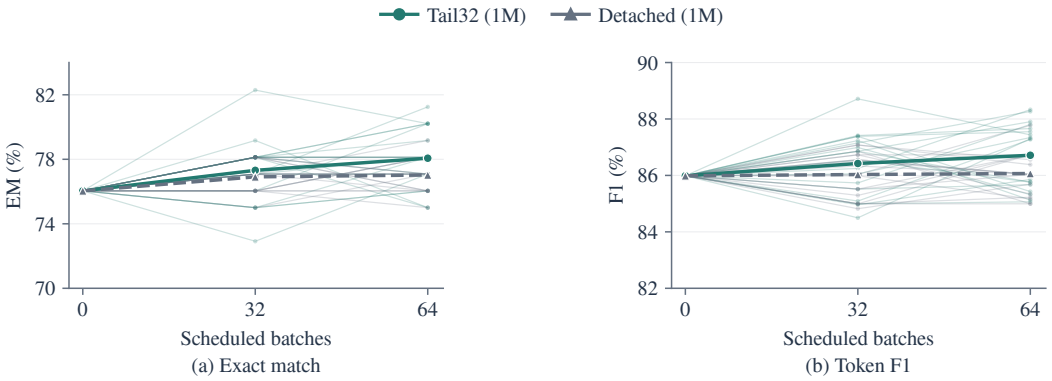


Figure 10: **Qwen QA learning at 1M tokens.** The 18-seed, 64-batch study uses a 32-token response cap; Tail32 and detached arms use complete 1M inputs. Strong lines show seed means and faint lines show every seed. The two panels report exact match and token F1; The 1M rows of Table 2 give the endpoints.

Endpoints and information controls. Figure 10 shows the learning curves; Table 15 gives endpoints and gains for intact, question-only and last-8K inputs. Both arms start at 76.04% EM and 85.99% F1. F1 gains are $+0.73$ $[0.15, 1.31]$ for Tail32 and $+0.08$ $[-0.29, 0.45]$ for detached; the descriptive tail-minus-detached contrasts are $+1.04$ $[0.21, 1.88]$ EM and $+0.65$ $[0.17, 1.12]$ F1 points.

Table 15: **Intact inputs and information controls across the two training arms.** Cells give final scores (percent) and changes from initialization (percentage points), averaged over all 18 seeds. Last-8K keeps only the final 8,192 tokens of each original prompt.

Method	Metric	Intact	Question only	Last 8K
Tail32	EM	78.07 (+2.03)	19.33 (−1.50)	0.00 (0.00)
	F1	86.71 (+0.73)	30.41 (−1.51)	2.02 (+0.15)
Detached	EM	77.03 (+0.98)	20.31 (−0.52)	0.00 (0.00)
	F1	86.07 (+0.08)	30.97 (−0.95)	1.99 (+0.13)

Last-8K EM is always zero with undefined intervals, and approximately 93–94% of outputs reach the cap. Intact seed–question pairs show 116 error-to-exact and 81 exact-to-error changes for Tail32, versus 42 and 25 for LongStraw detached. The examples with the largest tail changes include "eight languages" to "eight" (18 improvements) and "Sky+ PVR" to "Sky+" (17 regressions); these span-boundary changes do not identify the full learning mechanism. Tail gold-first-token mass rises from 73.28% to 80.68% and mean output length falls from 6.45 to 5.49 tokens.

Resources and sensitivity. Tail32 and detached runs average 57.11 and 62.61 effective updates (ranges 50–62 and 60–64). Informative groups average 23.29 and 32.15%, and peak PyTorch allocated memory averages 125.46 and 125.18 GiB. Mean training costs are 20.46 and 19.98 GPU hours, excluding failed runs. Synchronized batch timings include sampling, scoring, backward, optimization, parameter consistency checks and rollout I/O; they exclude loading, evaluation, checkpoint archival, queueing and waiting. Prefix capture takes 15.69 and 15.73 hours on average, with mean per-run shares of 76.73 and 78.73% of training time. These times include blocked prefix computation and cache/parameter consistency checks but exclude subsequent input checks and cache clearing. The remaining 4.76 and 4.25 hours combine sampling, scoring, backward, optimization and other overhead; separate recomputation and transfer costs are unmeasured.

Prespecified 20,000-draw paired bootstrap sensitivities give Tail32 EM-gain intervals of $[-3.70, 7.93]$ points when resampling seeds and articles and $[-7.41, 11.52]$ when archives are also resampled. Per-archive gains are $-6.25, +7.18, +4.40, +2.78$ points; every leave-one-archive-out gain is positive, ranging from $+0.31$ to $+4.78$ points.

Numerical validation scope. All 36 full-input runs provide prespecified endpoints. Native comparisons cover 32K tail and detached execution (Appendix J) and extend to 512K (Appendix C.2); Appendix G tests saved-tensor placement and checkpointed autograd. At 1M, native autograd with CPU offload runs out of memory, whereas LongStraw completes the update.

I.1 Two- and Four-Million-Token QA with a 512-Token Response Budget

These studies train Qwen3.6-27B NF4 with rank-16 LoRA on four H200s at CP4 for 2M and eight H200s at CP8 for 4M. Inputs contain exactly 2,097,152 or 4,194,304 tokens, including a 256-token question segment; a 32-token prompt tail remains differentiable. Static YaRN uses factors 8 and 16, respectively, with original maximum position 262,144 and configured maxima 2,097,664 and 4,194,816. Prefix capture uses query-parallel FA2, and the response reverse sweep processes grouped-query attention heads in chunks. These kernels differ from the 1M study. The 4M study appends 14,126 HotpotQA training paragraphs, without questions or answers, to the 1M distractor pool, for 29,047 paragraphs and 4,456,549 tokens.

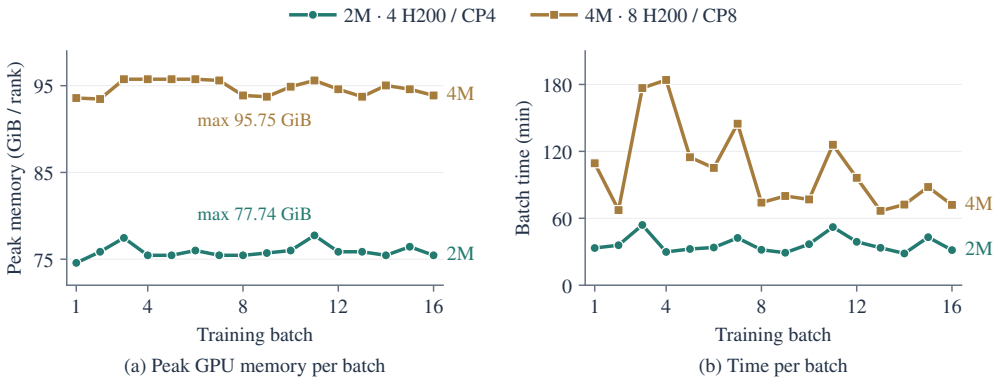
In each study, training seed 4101 visits all 128 fixed training questions once in 16 batches of eight questions, with $G = 4$ responses per question sampled at temperature 1 from the full softmax. F1 rewards supply group-standardized advantages for the unclipped policy-ratio objective without KL; zero-variance groups remain in the batch denominator. AdamW uses learning rate 10^{-5} and zero weight decay, with gradient-norm clipping at 1. Every batch produces a parameter-changing update, for 16 updates and 512 sampled responses. Prefixes are recaptured with the current policy for each batch. Both training and greedy evaluation allow up to 512 generated tokens, with natural EOS, no minimum response length, and thinking disabled. At 2M, one training response reaches the cap; the initial and final evaluations have zero and one cap hits. At 4M, 15 of 512 training responses reach the cap, with eight initial and four final evaluation cap hits.

The initial and step-16 checkpoints answer the same 96 held-out questions using identical full inputs. Training and evaluation question IDs, source articles and supporting passages are disjoint. The evaluation questions share four archives of 24 questions each. All 192 predictions in each study are scored with the original SQuAD-style EM/F1 normalization: lowercase, punctuation and article removal, whitespace normalization, and the maximum score over reference answers. Checkpoints preserve adapter parameters, optimizer state and rank-local random-number states.

Training time and memory. Figure 11 reports the same 16 effective updates used in the learning results. Each memory observation is the maximum rank-local allocated peak within one batch; the run maximum is 77.7411 GiB at 2M and 95.7450 GiB at 4M. Synchronized batch times sum to 35,225.87 s (9.78 h) and 99,271.16 s (27.58 h), with medians of 33.69 and 92.17 minutes. Timing includes prefix capture, natural-EOS generation, scoring and replay, backward, gradient synchronization, optimizer work and within-batch diagnostics. Model loading, initial and final evaluation, checkpoint writes outside a batch, and failed runs are excluded. The 16 observations are sequential batches of completed training, not independent training replicates.

2M paired results. Table 2 reports EM rising from 56.25% to 57.29% and F1 from 67.11% to 69.37%. The paired changes are $+1.04$ percentage points in EM and $+2.27$ in F1. Percentile 95% intervals from 200,000 paired-question resamples are $[0.00, 3.13]$ and $[0.20, 5.10]$ points, respectively. EM improves on one question and declines on none; F1 improves on five, declines on two, and is unchanged on 89. F1 rises in all four archives, by 0.64, 7.05, 0.21 and 1.17 points. Mean response length falls from 19.01 to 17.34 generated tokens.

4M paired results. After all 16 updates, EM increases from 27.08% to 33.33% ($+6.25$ points) and F1 from 38.22% to 43.10% ($+4.88$ points). EM improves on seven questions, declines on one and is unchanged on 88; F1 improves on 15, declines on 13 and is unchanged on 68. EM rises in all



(a) Peak GPU memory per batch

(b) Time per batch

Figure 11: **Per-batch cost of the 2M and 4M QA runs.** (a) Peak allocated GPU memory and (b) time for each of the 16 consecutive training batches of each run, including generation (512-token cap). Lines connect batch order; memory is the maximum allocation across ranks. Appendix I.1 gives the protocol.

four archives, by 8.33, 4.17, 4.17 and 8.33 points. All 192 predictions and 512 training rewards are independently rescored with the fixed metric implementation. Table 2 reports both endpoints.

I.2 Learning on RULER Multi-Value Retrieval

We train on the RULER `niah_multivalue` task (Hsieh et al., 2024), which asks for four seven-digit values associated with one key in a long essay. A separate development-set comparison uses 16 examples per task: greedy reference recall is 81.25% for multi-value retrieval, 97.92% for frequent-word extraction, and 100% for default variable tracking. Four sampled responses per development question show reward variation in 12/16 multi-value groups and 11/16 frequent-word groups. Multi-value retrieval is selected before generating its training and evaluation splits; these development scores are not estimates of training improvement on the independent evaluation split.

Fixed data and update protocol. The official generator and answer-prefix template are preserved. Training and evaluation each contain 64 newly generated examples, using data seeds 17101 and 17102, respectively; the development seed is 16101. Complete prompt/token sequences, queried keys, and gold values are disjoint across the three splits, although they use the same essay distribution. The 128K length budget includes up to 128 generated tokens. No prompt or gold answer is truncated, and all four gold values fit within 31 tokens.

One run starts from the same untrained adapter checkpoint used for the development-set comparison (initialization seed 4101), with NF4 Qwen3.6-27B, BF16 compute, FP32 adapters, YaRN4, and one H200. It visits all 64 training examples once in fixed order, sampling $G = 4$ responses with temperature 1 and the full softmax. Response seeds are $17103 + 4(k-1) + j$ for group k and response $j \in \{0, 1, 2, 3\}$. Reward is unrounded official four-reference recall; advantages subtract the group mean and divide by population standard deviation. The objective uses an unclipped, sequence-normalized policy ratio, no KL term, and a log-ratio safety clamp to $[-20, 20]$. AdamW uses learning rate 10^{-5}, zero weight decay, and gradient-norm clipping at 1.

For a prompt of length P, the gradient boundary is $32\lfloor P/32 \rfloor - 32$, so the differentiable tail spans 32–63 actual prompt tokens rather than exactly 32. The reverse sweep includes the full generated response, its first token and EOS; storage padding is masked. Each group captures the current policy's prefix. Zero-variance groups are retained without redraw or an optimizer step. The run completes all 64 groups with 26 effective updates and 38 zero-variance groups. Checkpoints are saved after groups 0, 1, 16, 32 and 64; the evaluated endpoint is fixed at group 64 before any independent test prediction is generated or evaluated.

Independent endpoint comparison. The initial and final checkpoints answer the same 64 held-out questions with greedy decoding, identical complete inputs, and the same 128-token output cap. Official scoring counts gold strings found after postprocessing. The four-reference all-hit rate counts

questions with recall 1; neither metric requires an exact match to the entire generated text or penalizes extra text and numbers. Table 16 gives both endpoints.

Table 16: **Fixed 128K multi-value retrieval: answer and question counts.** 64 questions, four references each, and one training run. Scores are percentages; changes are percentage points. Intervals resample paired test instances, conditional on this trained model.

	Correct / total		Score (%)		
Metric	**Initial**	**Trained**	**Initial**	**Trained**	**Change [95% CI]**
Reference recall	228/256	255/256	89.06	99.61	$+10.55$ $[3.52, 18.75]$
Four-reference all-hit	57/64	63/64	89.06	98.44	$+9.38$ $[1.56, 17.19]$

The 95% percentile intervals use 10,000 paired-instance bootstrap resamples. Seven initially failed questions recover all four answers: six initial outputs contain only a colon and one contains only ": 1.", each followed by EOS. One previously complete answer instead returns three of four values and then EOS, within the output cap. The gain thus reflects fewer premature answer terminations and better answer coverage; it does not establish stronger retrieval.

Synchronized training groups consume 10,353.6 seconds (2.88 GPU hours), including prefix capture, sampling, scoring, backward, and optimization but excluding initial model loading and checkpoint I/O. Peak allocated GPU memory is 41.00 GB. Mean full-input evaluation time is 52.31 seconds per question initially and 55.26 seconds after training, including capture and generation. The single-key transfer task is saturated (Appendix I.3), preventing measurement of learning gains.

I.3 RULER Checkpoint Transfer

We report evaluations on 500 fixed examples at each of 128K and 256K on the RULER `niah_multikey_1` task (Hsieh et al., 2024): official data seed 42, essay distractors, four word keys with seven-digit values, and one queried key. Input and answer-prefix bytes are preserved without chat templates or truncation. Greedy decoding allows 128 tokens within the length budget. Each process uses one H200, sharing NF4 weights, tokenizer, full-input runtime and YaRN4.

We evaluate the untrained base model and the seed-4101 Tail32 and detached checkpoints after batch 64 of the 1M study (Appendix I); all three receive complete RULER inputs.

All six conditions (three checkpoints at 128K and 256K) score 100% (500/500 questions). Their 3,000 nonempty predictions use official postprocessing and string-match scoring. These saturated scores establish retention on this subtask, not learning gains.

J Numerical Validation

Comparisons separate objective scope from implementation: native full-prompt autograd (A), native detached-prefix autograd (B), and LongStraw at the same gradient boundary (C). A–B changes the objective; B–C tests agreement with an independent native reference. All match base weights, NF4/BF16 computation, rank-16 adapters, tokens, positions, masks, advantages and optimizer state. Native caches update out of place from cloned boundary states. Both paths use matching GDN convolution and official FA2 backward; 32K, distributed, 128K and 512K comparisons also match FP32 recurrent state, fixed FLA launches and deterministic FA2 backward.

All 992 named adapter gradients and AdamW parameter changes are compared, together with selected log probabilities before and after the update. For the controls of Table 17, the prespecified relative L_2 limits are 0.05 for gradients and updates, and the maximum absolute score-error limit is 0.005; the 128K and 512K comparisons set no acceptance threshold. Table 17 reports the resulting errors. The 32K CP1 comparison also gives exact agreement. A one-token differentiable-tail case at $P = 65$, $R = 31$, $G = 2$ reproduces gradients, updates, and post-update scores exactly.

The distributed control shards persistent K/V pages and gathers one layer for official FA2. Each rank computes the global query result, retains its suffix gradients and sums current K/V gradients across ranks; comparison precedes averaging. The uneven-page case assigns twelve 64-token prefix pages to eight ranks, with 256-token prefix blocks and 64-token response blocks, testing uneven

Table 17: **Matched-objective numerical errors across distributed layouts.** Gradient and AdamW-update errors are relative L_2; score errors are maximum absolute log-probability differences. All rows use $G = 2$ and compare all 992 named adapter tensors. The uneven-page case also tests execution with two separate response-backward blocks.

Control	CP ranks	P	R	Gradient	AdamW update	Pre/post score
Single rank	1	64	32	0	0	0 / 0
Distributed	2, 4, 8	512	64	0	0	0 / 0
Uneven pages	8	768	128	1.9390×10^{-10}	1.8605×10^{-10}	0 / 0

ownership and two response-backward blocks. All ranks agree on gradients and updated parameters. This reference has different temporary memory costs from the optimized distributed backend.

J.1 Bounded-Length Update Cost

At 32K on one H200, native shared-prefix execution is faster than LongStraw (median 6.74 versus 9.32 seconds) and uses 0.22 GiB less peak GPU memory (Table 18).

Table 18: **Matched 32K update cost.** Three-repeat medians [ranges]; ratios and GPU changes are relative to native shared. Memory: peak GPU allocation and host RSS with allocator retention. CPU staging trades GPU memory for time and host memory.

	Update time		GPU memory		Host memory
Implementation	Step (s)	Ratio	Peak GiB	Change GiB	Range GiB
Native shared	6.74 [6.66, 6.99]	1.00×	33.62	0.00	6.18–7.98
Native + CPU staging	15.10 [14.57, 15.73]	2.24×	26.62	−7.00	11.51–11.54
LongStraw FA2	9.32 [9.14, 9.42]	1.38×	33.84	+0.22	6.18–8.37

Both native-detached arms and LongStraw use $R = 128$, $G = 2$, identical NF4 weights, FP32 recurrent state, adapters, AdamW, and deterministic FA2. Native execution shares immutable prefix K/V and copies mutable recurrent state per response. All three implementations reproduce the same gradients and updates. Three warm repeats rotate arm order and time capture, backward, and optimization, excluding archival and post-step scoring.

K Position Encoding

Position handling follows the pretrained backbone. The Qwen and GLM checkpoints use RoPE with default ranges of 262,144 and 1,048,576 positions (Qwen Team, 2026; zai-org, 2026). We use static YaRN factor 4 for Qwen 1M (maximum position 1,048,608), and factors 8 and 16 for 2M and 4M; GLM 2M uses factor 2 for 2,097,152 positions (Peng et al., 2024).

NoPE (no explicit positional encoding) provides another design choice. Kimi Linear uses NoPE in its full-attention MLA layers and relies on recurrent KDA layers for positional information (Kimi Team, 2025). Position encoding is thus a backbone choice; LongStraw organizes state storage and backward computation for the selected model.