

LongStraw: Long-Context RL Beyond 2M Tokens under a Fixed GPU Budget

Changhai Zhou^{1,2}, Kieran Liu¹, Yuhua Zhou¹, Qian Qiao¹, Jun Gao¹, Harry Zhang¹, Irvine Lu¹, Nolan Ho¹, Lucian Li¹, Andrew Lei¹, Cleon Cheng¹, Steven Chiang¹, Yihang Zeng¹, Di Zhang¹, Rio Yang¹, Kaijie Chen¹, Andrew Chen^{1,*}, Pony Ma^{1,*}, Weizhong Zhang², Cheng Jin²

¹MindLab, ²Fudan University

Abstract

A growing gap has emerged between the context lengths supported at inference time and those used during RL post-training. Inference systems are approaching million-token contexts, whereas post-training workloads often remain at 256K tokens or below and rely on length generalization at deployment. The gap matters particularly for AI agents, whose observations, tool outputs, documents, and prior decisions accumulate over long trajectories. Unlike inference, training must score and backpropagate through multiple responses conditioned on the same history. Quadratic attention computation and long-lived backward states make GPU memory a major bottleneck for extending training context.

We present **LongStraw**, an architecture-aware execution stack for million-token RL post-training under a fixed GPU budget, instantiated with Group Relative Policy Optimization (GRPO). LongStraw evaluates the shared prompt once without automatic differentiation, retains only the model-specific state required by later tokens, and replays short response branches one at a time under autograd. This reduces the live training graph from the full prompt and response sequence to a single response branch, trading additional replay time for lower GPU memory usage. We implement LongStraw for two substantially different model families: the hybrid recurrent and full-attention Qwen3.6-27B, and the compressed-attention mixture-of-experts GLM-5.2.

On eight H20 GPUs, LongStraw completes grouped scoring and response backward for Qwen at 2.1M positions with group sizes of 2 and 8. Increasing the group size from 2 to 8 adds only 0.21 GB of peak allocated memory. A separate stress test extends the execution envelope to 4.46M positions. On 32 H20 GPUs, we validate the end-to-end LongStraw execution path for a 2.1M-token prompt across all 78 layers of GLM-5.2.

These results show that state lifetime and physical ownership are key determinants of the practical context limit of RL post-training. The current experiments establish execution capacity rather than complete training correctness because the captured prompt state is detached and some distributed forward and gradient composition paths remain incomplete. By reducing reliance on ever-larger GPU clusters, LongStraw lowers the hardware barrier to long-context training and enables more researchers and smaller teams to explore this direction under limited accelerator budgets.

Correspondence: Changhai Zhou: chzhou25@m.fudan.edu.cn

Andrew Chen: andrew@mindlab.ltd

Pony Ma: pony@mindlab.ltd

Code: <https://github.com/MindLab-Research/longstraw>

Date: July 2026

Contents

1 Introduction	3	8 Execution Receipts and Trace Evidence	23
1.1 Report Roadmap	4	8.1 Runtime Ownership and Pinned Rerun Stack .	24
2 GRPO Training Dependency Graph	4	8.2 Qwen Receipts	24
2.1 Branches and Ordering Constraints	5	8.3 GLM Terminal Manifest	24
2.2 Conditional Replay and the Missing Gradient Term	6	8.4 Trace Inventory	25
2.3 Four Levels of Evidence	6	8.5 Memory Accounting	25
3 Architecture Anatomy and Bottleneck Sources	6	8.6 Semantic Claim Matrix	25
3.1 Qwen: Dense FFNs with Hybrid Token Mixing	7	9 Fixed-Budget Systems Lessons	26
3.2 GLM Attention: MLA, DSA, and IndexShare .	8	9.1 Fixed-Budget Capacity Comes from Lifetime, Not Sparsity Alone	26
3.3 GLM Feed-Forward Path: Dense then MoE . .	8	9.2 Physical Ownership Is Part of the Algorithm .	26
3.4 What the Long Prompt Must and Need Not Retain	9	9.3 Dense and MoE Move the Peak to Different Places	26
4 LongStraw: Long-Context Execution Design	9	9.4 Context and Expert Parallelism Are Orthogonal	27
4.1 Capture Once, Replay the Suffix	9	9.5 Forward Fidelity, Update Consistency, and Gradient Parity Differ	27
4.2 State Inventory and Ownership	10	9.6 Group Scaling Is a Scheduling Result	28
4.3 Device Placement and Layer Staging	10	9.7 Eight H20s Carry Qwen to a 4.25M Context Envelope	28
4.4 Whole-Layer Checkpointing	11	9.8 The Claim Is a Fixed-Budget Envelope, Not a Context Record	28
4.5 Parallel Layout	11	10 Related Work	29
4.6 Adapter and Optimizer Scope	11	10.1 Scale-Out Long Context and the Fixed-Budget Axis	29
5 Qwen: Dense Hybrid Replay within an Eight-H20 Budget	12	10.2 Relationship to MinT	29
5.1 Hybrid Layer Anatomy and the Prompt Boundary	12	10.3 Memory-Efficient and Distributed Attention . .	29
5.2 From Logical Shards to Physical Pages	13	10.4 MLA, Sparse Attention, and Index Reuse . . .	30
5.3 Global Full-Attention Forward Composition . .	13	10.5 MoE Training and Multidimensional Parallelism	30
5.4 Four-Block Response Replay	14	10.6 Distributed Training State and Optimizer Sharding	30
5.5 The Fixed Eight-H20 Execution Envelope . . .	15	10.7 Activation Checkpointing and PEFT	31
5.6 Why the Receipt Is Not Yet a Coherent CP8 Update	15	10.8 Adapter Serving and Inference Infrastructure .	31
6 GLM: Paged MLA/DSA Replay within a 32-H20 MoE Budget	16	10.9 GRPO Systems	31
6.1 Layer Anatomy: MLA/DSA Before Dense or Routed FFNs	16	11 Conclusion	32
6.2 TP1/CP32/EP32 Assigns Two Different Kinds of Ownership	16	12 Limitations and Validation Roadmap	32
6.3 Zigzag Context Pages within the 32-H20 Route	17	12.1 Distributed Gradient Composition Is Incomplete	32
6.4 Stored Tensor Contract and Derived Residency	17	12.2 The Historical GLM Receipt Uses CP-Local Response Attention	33
6.5 One Replayed Layer: State, Index, Attention, and MoE	18	12.3 The Prompt-State Gradient Is Detached . . .	33
6.6 Whole-Layer Checkpointing and the Recorded Backward	19	12.4 The Workload Is an Execution Probe	33
6.7 Why the DSA Operator in the Accepted Receipt Is Local, Not Full Context	19	12.5 Resource and Reproducibility Gaps	33
6.8 Gradient Finalization Audit: A Terminal Call Is Not an Update	20	12.6 Validation Order	33
7 Making the GLM GRPO Path Fit a Fixed 32-H20 Budget	21	Appendices	37
7.1 Stage I: Full-Graph Bottleneck Localization . .	21	A Model and Run Configuration	37
7.2 Stage II: Prefix-State Scaling	22	B GLM Page Mapping and State-Size Derivation	38
7.3 Stage III: Single-Layer Differentiable Replay . .	22	C Representative GLM Trace Contract	38
7.4 Stage IV: All-Layer Architecture Closure . . .	22	D Distributed-Gradient Audit	38
7.5 Stage V: Parallel Ownership and CPU Pages . .	22	E Detailed GLM Capacity Progression	39
7.6 Stage VI: Fixed-Budget Optimizer-Call Canary	23	F Qwen 4.25M Replay within Eight H20s	39
7.7 Stage VII: Fresh Grouped Execution	23	G Acknowledgements	41

1 Introduction

AI agents are moving beyond one-shot answers toward using tools, inspecting code and documents, and acting over long trajectories. ReAct formalizes this interaction as a sequence of reasoning, actions, and observations (Yao et al., 2023a), while LongCat-Flash-Thinking trains agents over long, multi-turn tool trajectories (Meituan LongCat Team, 2026). For these agents, context carries evidence, environment observations, tool outputs, and earlier decisions into the next action.

Long-context inference and post-training use memory differently. An inference server can prefill a prompt, cache the state used for decoding, and discard the forward graph (Kwon et al., 2023). Post-training must score several responses and backpropagate through them. Group Relative Policy Optimization (GRPO) compares responses that share a prompt through group-relative advantages (Shao et al., 2024). Each response may be short, but its score still depends on the full prompt and its cached state.

Existing techniques reduce important parts of this cost, but they do not by themselves make a fixed-GPU GRPO run fit. Memory-efficient attention reduces the workspace of an attention layer (Rabe and Staats, 2021). FlashAttention improves the data movement of exact attention (Dao et al., 2022). LoRA reduces the number of trainable parameters (Hu et al., 2021), while QLoRA also reduces the storage cost of the base model (Dettmers et al., 2023). The prompt graph, response graphs, cached state, and distributed communication still compete for the same device memory.

Large accelerator fabrics can extend sequence length by distributing this work more widely. Ring Attention reports 4.096M-position training for a 7B model on 32 A100 GPUs (Liu et al., 2023). DeepSpeed-Ulysses studies one-million-token training while scaling to 256 A100 GPUs (Jacobs et al., 2023). ByteScale reports a 2M LLaMA-7B case on 1,024 GPUs (Ge et al., 2025), and USP combines ring and all-to-all sequence parallelism (Fang and Zhao, 2024). These systems establish the scale-out route. This report asks a complementary question: *how far can a GRPO execution path go when the GPU count stays fixed?*

LongStraw answers this question by separating the long prompt from the short response computation. It evaluates the shared prompt once without automatic differentiation, stores the information needed to condition later tokens, and then processes one response at a time. Gradients from the response members are accumulated before the optimizer is called. Serial replay increases elapsed time, but it avoids keeping the prompt graph and every response graph live at the same time.

The stored information follows the model architecture. Qwen3.6-27B combines recurrent layers with full-attention layers (Qwen Team, 2026); LongStraw keeps the recurrent state and context-sharded key/value pages needed by later tokens. GLM-5.2 uses compressed attention state, sparse attention indices, and routed experts (zai-org, 2026). Its implementation moves the prompt state to CPU memory and stages one decoder layer at a time. Figure 1 shows the shared execution schedule and where the two implementations differ.

LongStraw runs inside the training side of MinT (Mind Lab, 2026). MinT manages model workers, adapter revisions, and the outer policy transaction; LongStraw manages the prompt state and response work inside one long-context transaction. The two systems therefore operate at different levels of the same training stack.

This execution boundary changes what is differentiated. Both implementations detach the stored prompt state from the response backward pass. The Qwen path does not synchronize every key/value-related gradient across context-parallel ranks. The GLM path uses attention keys from each rank’s local prompt shard and skips the usual cross-rank gradient reduction. We therefore report completed execution and local optimizer calls separately from claims about a correct distributed update or learned policy quality.

Contributions. This report makes three contributions:

- It presents a shared-prompt execution design that stores prompt state once and serializes response work under a fixed GPU budget.
- It implements that design for two different model structures: a Qwen hybrid recurrent/attention stack and a GLM compressed-attention/MoE stack.
- It reports the completed paths, timing, and memory measurements together with the distributed operations that remain unverified.

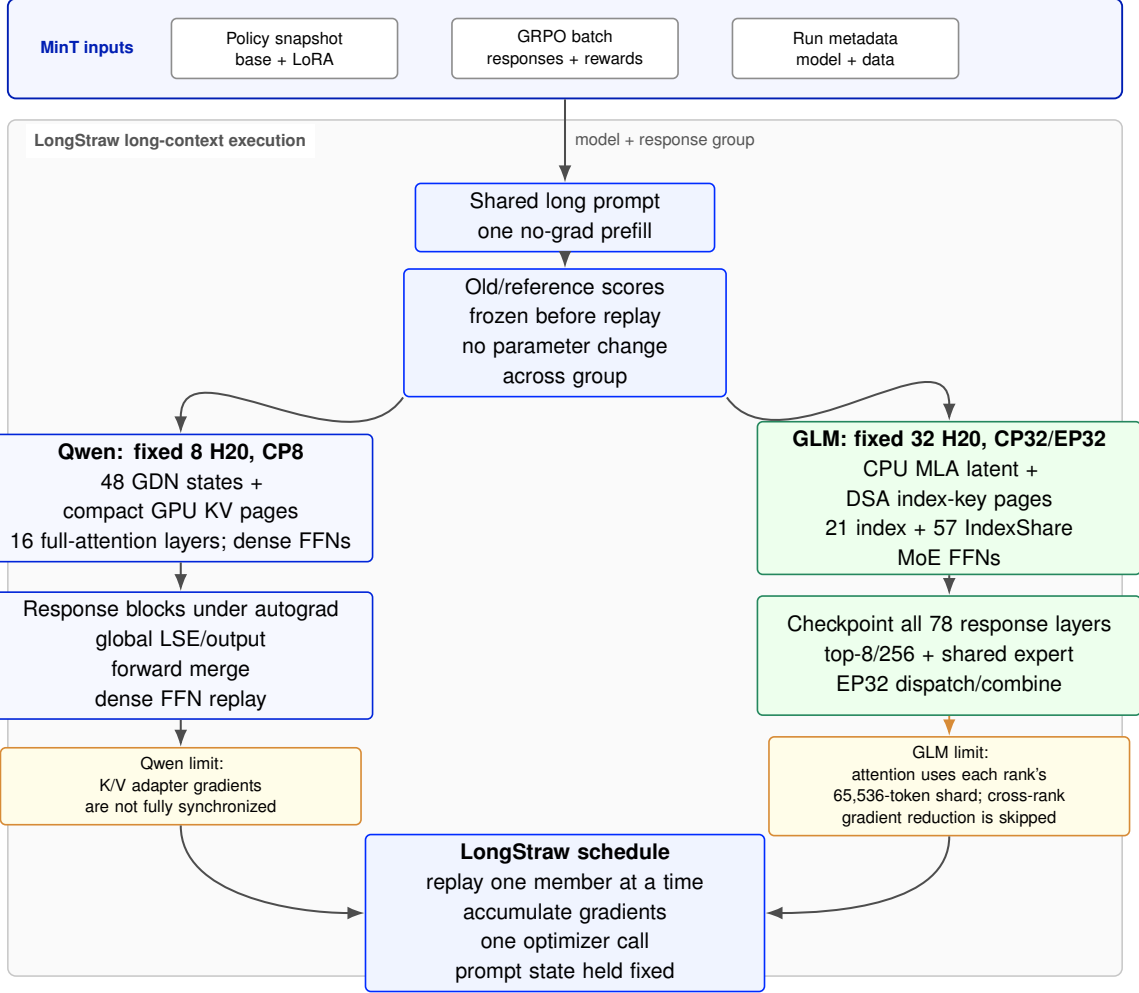


Figure 1 LongStraw execution path. The training runtime supplies a model snapshot and a group of responses. LongStraw evaluates the shared prompt once, stores model-specific state, then replays one response at a time and accumulates gradients. Qwen keeps recurrent state and sharded KV pages; GLM keeps CPU MLA/DSA pages and checkpoints its response layers. The orange boxes state the distributed reductions that the measured paths do not complete.

1.1 Report Roadmap

The report first defines the training dependency and model structures, then describes the Qwen and GLM implementations. The remaining sections present the fixed-budget measurements, explain the main memory and communication costs, and state the limits of the current execution paths.

2 GRPO Training Dependency Graph

The systems problem begins with the update graph, not with a particular attention kernel. Let a prompt contain P tokens, group member i contain R_i scored response tokens, and the group contain G responses. For old policy π_{old} , current policy π_{θ} , and normalized advantage A_i , define

$$\rho_{i,t}(\theta) = \exp(\log \pi_{\theta}(y_{i,t} \mid x_{1:P}, y_{i,<t}) - \log \pi_{\text{old}}(y_{i,t} \mid x_{1:P}, y_{i,<t})). \quad (1)$$

The clipped policy term is

$$\mathcal{L}_{\text{policy}} = -\frac{1}{G} \sum_{i=1}^G \frac{1}{R_i} \sum_{t=1}^{R_i} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) A_i), \quad (2)$$

with a tokenwise reference-policy KL term weighted by β . The clipped ratio surrogate follows PPO (Schulman et al., 2017); group-relative advantages, member normalization, and the reference-policy penalty follow the GRPO objective (Shao et al., 2024). This is the member-normalized GRPO term implemented by the audited paths. The contribution of this report is how its conditional log-probabilities are executed when the context length exceeds two million positions under a fixed accelerator allocation without adding devices.

2.1 Branches and Ordering Constraints

A grouped update has five logically different phases.

1. **Prompt capture.** Evaluate the shared prompt without autograd and retain the architecture-specific conditional state.
2. **Pre-step scoring.** Evaluate old-policy and reference-policy response log-probabilities before the corresponding policy backward. Parameters remain fixed through the group. GLM materializes both old score sets first; Qwen performs old/reference scoring and policy replay member by member.
3. **Advantage construction.** Convert group rewards into the advantages used by Equation 2.
4. **Policy replay.** Rebuild one short response graph at a time, backpropagate its loss, and accumulate gradients into the same adapter.
5. **Optimizer transaction.** Synchronize accumulated gradients, step once after all G members, and clear gradients.

The ordering is load-bearing. Stepping between group members changes both the importance ratio and the prompt state. Holding all response graphs makes activation memory scale with G , whereas serial replay makes group cardinality primarily a scheduling and time dimension. This is a systems statement, not a claim that G is statistically irrelevant: G still defines reward normalization and the GRPO advantage set. Figure 2 contrasts the full-sequence and captured-state graphs.

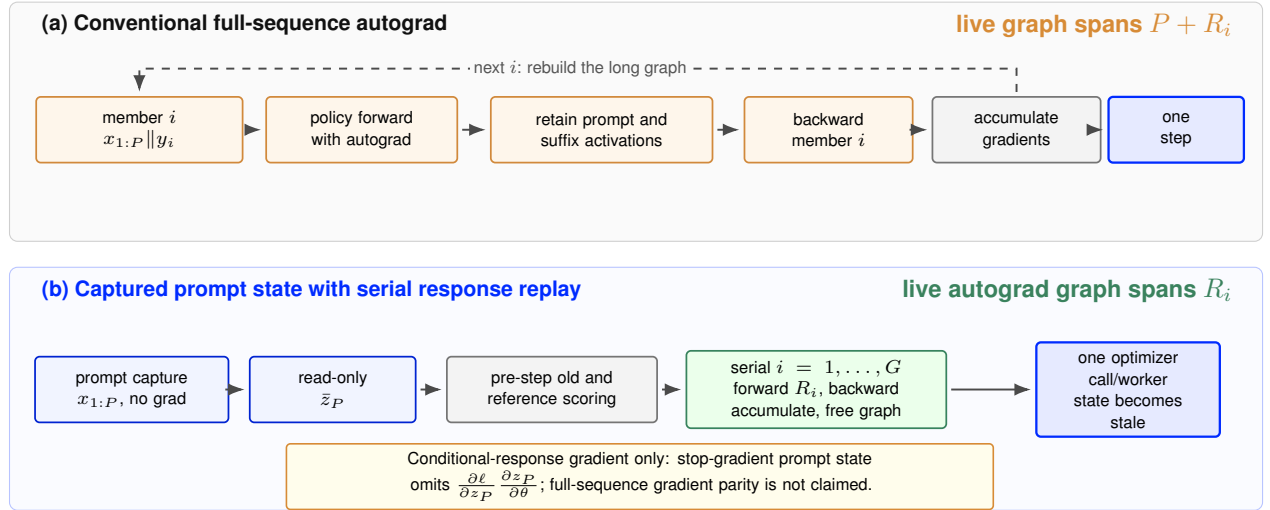


Figure 2 Changing the graph boundary, not the GRPO objective. (a) Conventional autograd retains prompt-dependent activations in each member graph. (b) The reported schedule captures a read-only prompt state, computes old/reference scores under unchanged pre-step parameters, and replays and frees one response graph at a time, accumulates G local gradients, and calls each worker optimizer once. Serial replay bounds live policy-autograd activations by one response, although group-indexed inputs and frozen scores still grow with the supplied group. It does not recover the gradient through the captured prompt state.

The acceptance runs use supplied deterministic responses and rewards. They exercise prompt capture, frozen scoring, live policy backward, group accumulation, and optimizer execution, but exclude online generation and reward-model execution. The report therefore covers only the executed training graph.

2.2 Conditional Replay and the Missing Gradient Term

Let $z_P(\theta)$ denote the complete model state after processing the prompt. A conventional full-sequence loss differentiates both its explicit response computation and the parameter dependence of the prompt state:

$$\nabla_{\theta} \ell(\theta, z_P(\theta)) = \left. \frac{\partial \ell}{\partial \theta} \right|_{z_P} + \frac{\partial \ell}{\partial z_P} \frac{\partial z_P}{\partial \theta}. \quad (3)$$

The implementations in this report store $\bar{z}_P = \text{stopgrad}(z_P(\theta))$ and compute only the first term. An exact attention reduction inside response replay can preserve the conditional response operator, but it cannot reconstruct the second term in Equation 3. “Exact attention” and “full-sequence gradient equivalence” are therefore different claims.

The distinction also defines state validity. A state captured from parameters θ_k may condition every response in update k , provided the parameters do not change between branches. After the optimizer produces θ_{k+1} , that state is stale. A repeated training loop must recapture the prompt or use an explicitly analyzed state-state approximation. The receipts stop after worker-local optimizer calls and provide no repeated-loop evidence.

2.3 Four Levels of Evidence

We use four progressively stronger tests. Distributed-update consistency remains separate from full-sequence parity because a correct distributed forward may still leave replicated parameter gradients rank-local.

Execution capacity. Did every requested score, backward, collective, and optimizer event complete on every rank with finite values? This is established by terminal logs and rank-local traces.

Response-operator fidelity. Does distributed replay compute the model-defined conditional response operator? Qwen reaches this level for full-attention layers through a global CP8 merge with BF16 numerator reduction. The GLM fallback does not: sparse selection remains local to each CP shard.

Distributed-update consistency. Are all sharded gradient contributions reduced to the correct parameter owner, and do replicated adapters remain identical after the optimizer call? The Qwen probe all-reduces dQ but not the local dK/dV contributions to replicated projection adapters. Its eight AdamW instances step independently. The current GLM resident path calls backward outside the normal Megatron schedule and skips `finalize_model_grads`; CP-replicated non-expert adapters therefore step from unreduced gradients. Neither path reaches this level.

Full-gradient parity. Do all trainable gradient shards and optimizer deltas match a conventional full-sequence reference? Neither path has completed this test. It requires a shorter context at which both the conventional and replay implementations fit, followed by parameter-by-parameter comparison.

An execution receipt passes when old scores are frozen, every group member produces a live backward, local gradients accumulate, each worker issues exactly one optimizer call, values remain finite, and all ranks terminate. This is a systems result, not evidence for stronger semantic levels.

3 Architecture Anatomy and Bottleneck Sources

The two models differ along two independent axes. The feed-forward axis is dense versus MoE. It determines parameter residency, token routing, and the shape of activation buffers. The token-mixing axis is GDN/full attention versus MLA/DSA; it determines retained prompt state and response-time collectives. Model-level labels such as “dense”, “MoE”, or “sparse” hide this separation.

Figure 3 opens both decoder stacks at the level needed by the training runtime. The diagram shows counts and ownership rather than implying that the two models share one layer template.

Table 1 The two model axes and the state that crosses the detached prompt boundary. Dense/MoE describes the FFN; full/GDN versus MLA/DSA describes attention.

Model	FFN topology	Attention topology	Durable prompt state	Response-time collectives
Qwen3.6-27B	64 dense gated FFNs	48 GDN + 16 full-attention layers	GPU GDN state + compact CP-sharded KV pages	Global CP8 forward merge; K/V adapter reduction missing EP32
GLM-5.2	3 dense + 75 MoE; 256 routed, top-8 + 1 shared	78 MLA/DSA; 21 index + 57 IndexShare layers	CPU CP-sharded MLA latent pages + index-layer DSA key pages	dispatch/combine; DSA CP-local; CP grad finalize skipped

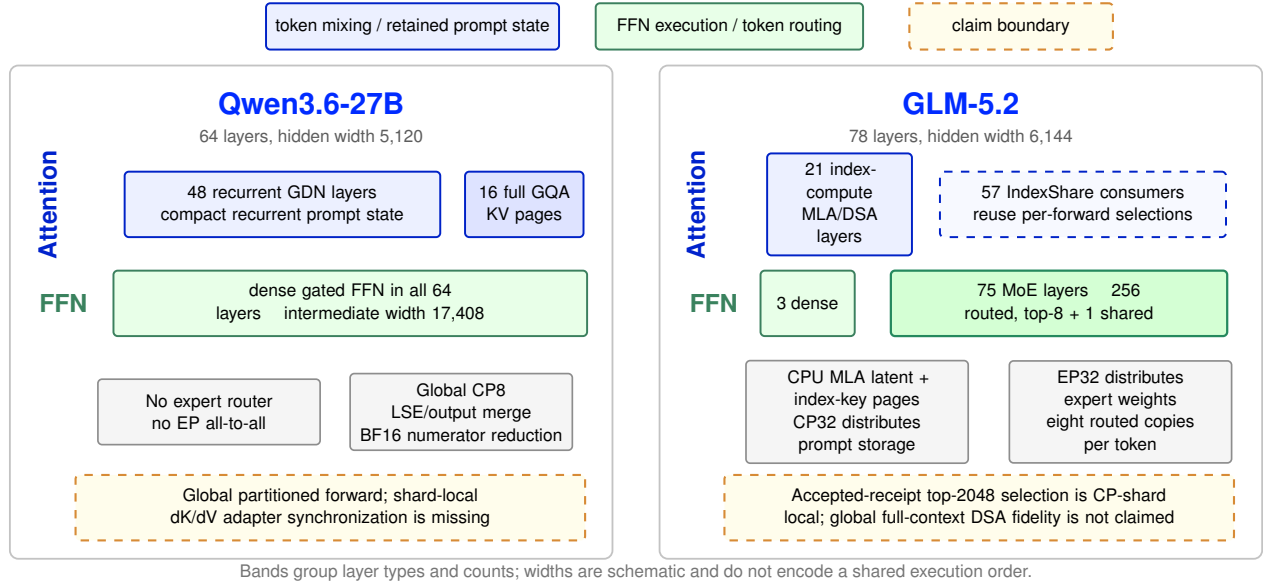


Figure 3 Two independent architecture axes. Blue bands encode token mixing and retained prompt state; green bands encode FFN execution and routing. Qwen combines 48 recurrent GDN and 16 full-attention layers with dense FFNs. GLM combines 21 index-computing and 57 IndexShare layers with three dense and 75 MoE FFNs. The orange boxes preserve the semantic boundary of each receipt: Qwen establishes global forward partitioning but not coherent distributed adapter updates; the accepted-receipt GLM DSA fallback is local to each context-parallel shard.

3.1 Qwen: Dense FFNs with Hybrid Token Mixing

The inspected Qwen3.6-27B configuration has hidden width 5,120, 64 decoder layers, 48 `linear_attention` entries, and 16 `full_attention` entries (Qwen Team, 2026). The audited runtime instantiates the former with its recurrent GDN module. Gated DeltaNet supplies that module’s gated delta rule (Yang et al., 2025), while grouped-query attention shares fewer KV heads across a larger set of query heads (Ainslie et al., 2023). Every layer ends in a dense gated feed-forward network with intermediate width 17,408. A gated dense FFN applies the same three matrices to every token in the SwiGLU form (Shazeer, 2020),

$$F_{\text{dense}}(h) = W_2(\text{SiLU}(W_1 h) \odot W_3 h). \quad (4)$$

There is no token router and no expert all-to-all. Once the long-prompt FFN graph is detached, a response replay only invokes the same dense matrices for the short suffix.

The two token mixers expose different prompt-state scaling. A GDN layer carries a fixed-shape recurrent state across the prompt boundary. A full-attention layer carries key and value pages whose storage grows with P . Full attention also has a global semantic requirement: every response query must attend to pages owned by all context-parallel ranks. Qwen thus places most of the long-context storage problem in 16 layers, while the other 48 layers contribute compact recurrent state.

The reported Qwen implementation uses NF4 QLoRA with 116,727,808 trainable parameters. Quantized base weights reduce persistent model storage, but they do not by themselves remove prompt pages or the response activation graph. The working design is therefore built around physical page compaction and conditional replay, not only parameter quantization (Dettmers et al., 2023).

3.2 GLM Attention: MLA, DSA, and IndexShare

Multi-head latent attention (MLA) compresses per-token KV content into a latent representation (DeepSeek-AI, 2024a). The inspected GLM-5.2 configuration has hidden width 6,144 and 78 decoder layers. Its KV and query latent widths are 512 and 2,048, respectively (zai-org, 2026). Its sparse indexer has 32 heads of dimension 128 and selects at most 2,048 preceding positions for each query (zai-org, 2026). Following the DSA definition (DeepSeek-AI, 2025), an index score can be written as

$$I_{t,s} = \sum_{j=1}^{H_I} w_{t,j}^I \text{ReLU}(\langle q_{t,j}^I, k_s^I \rangle), \quad H_I = 32, \quad (5)$$

and the sparse response output is

$$u_t = \text{Attn}(q_t, \{c_s : s \in \text{TopK}(I_{t,:}, 2048)\}). \quad (6)$$

The 2,048 selected positions define the operator, not merely its memory layout. Faithful distributed replay requires global selection over the logical context and correct composition of the selected values.

Computing a new sparse index in every layer would repeat similar work. The inspected GLM configuration instead yields 21 index-computing layers and 57 IndexShare layers that consume a selection published by a nearby source layer (zai-org, 2026). This cross-layer reuse resembles the mechanism analyzed by IndexCache (Bai et al., 2026). It changes the runtime contract in two ways. First, only the 21 compute layers need durable prompt indexer-key pages. Second, the response forward must maintain a per-forward producer/consumer holder whose index schedule cannot span response branches or parameter versions.

3.3 GLM Feed-Forward Path: Dense then MoE

The inspected configuration assigns dense FFNs to the first three GLM decoder layers and 256 routed experts with top-8 routing plus one shared expert to the remaining 75 layers (zai-org, 2026). For token t ,

$$F_{\text{MoE}}(h_t) = F_{\text{shared}}(h_t) + \sum_{e \in \text{Top8}(g(h_t))} p_{t,e} F_e(h_t). \quad (7)$$

MoE sparsity reduces the number of experts evaluated for one token, but it creates a distributed parameter and data-movement problem. EP ranks hold different experts. The router expands one token into eight expert assignments, dispatches those rows to their owners, evaluates the selected expert paths, and combines the outputs (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022).

The parameter and activation scales explain why full-sequence GLM autograd is not made cheap by sparsity. An expert has intermediate width 2,048, so its three gated-FFN matrices contain

$$3 \times 6144 \times 2048 = 37,748,736 \quad (8)$$

weights. The 256 routed experts in one sparse layer contain approximately 9.66 billion weights, even though a token activates only eight of them. At a balanced 65,536-token CP shard, top-8 routing produces $65,536 \times 8 = 524,288$ expert-token rows. One BF16 buffer with shape $[524,288, 6144]$ occupies exactly 6 GiB before outputs, permutations, LoRA intermediates, or routing skew. A conventional beyond-2M-context graph can hold several such tensors across layers and backward phases.

Expert parallelism distributes the parameter working set, but it does not change the number of routed token copies. Context parallelism distributes attention state, but it does not place experts. Prior MoE folding work studies heterogeneous mappings across these parallel dimensions (Liu et al., 2025). In our run, folding CP32 and EP32 over the same ranks reduces separate process groups and matches the available 32 GPUs; the two dimensions still solve different ownership and communication problems.

3.4 What the Long Prompt Must and Need Not Retain

Both models still execute every prompt token through every layer. The next layer needs the output even when the current layer is sparse or recurrent. The capacity gain comes from tensor lifetime: dense FFN intermediates, MoE routes, expert-token permutations, attention scratch, and adapter activations from the prompt are allowed to die immediately. Only the conditional state required by future response tokens survives. For Qwen this is compact GDN and KV state. For GLM it is MLA latent pages, selected indexer-key pages, position/page metadata, and the rules needed to reconstruct IndexShare during each short replay.

4 LongStraw: Long-Context Execution Design

LongStraw is the execution stack developed in this report. Its design follows one rule: retain a tensor across the prompt boundary only when a later response token depends on it. The rule applies to physical allocations, not just logical tensor views. It also separates state required by the model operator from scratch and activations required only while producing that state.

Throughout this report, *budget-constrained* means a fixed accelerator count together with each device’s H20 memory limit: eight H20 GPUs for the Qwen path and 32 H20 GPUs for the GLM path. We report elapsed time and allocated GPU memory where the artifacts support them; a whole-transaction GLM peak is not available. Host-memory capacity, network traffic, energy, utilization, and monetary cost are not fully accounted. The resource claim is therefore fixed-device feasibility for the stated transaction, not a lowest-cost or state-of-the-art efficiency claim; those require matched baselines and full-system resource accounting.

4.1 Capture Once, Replay the Suffix

For update k , the runtime performs the following transaction.

Phase 1: Run $x_{1:P}$ under θ_k with autograd disabled. At each layer, save the model-specific prompt state and release transient hidden tensors, attention scratch, FFN activations, and MoE routing buffers.

Phase 2: Treat the completed prompt state as read-only. Materialize each response’s old/reference log-probabilities before its policy backward, without updates between group members. GLM freezes both old score sets before replay; Qwen uses a member-serial old/reference/policy schedule.

Phase 3: For each group member, rebuild the short current-policy response path under autograd. Reuse the same read-only prompt state, backpropagate the member loss, and immediately release that member’s graph.

Phase 4: After all G backwards, retain the accumulated local gradients and issue one optimizer call per worker. The captured state is now stale with respect to the worker’s updated parameters.

This schedule changes the dominant activation scale from $P + R$ to R , but prompt compute remains. Phase 1 still sends the full prompt through every decoder layer and retains architecture-specific state: full-attention pages, recurrent state, and DSA latent values plus index keys.

Nothing in the transaction is specialized to $G = 2$ or $G = 8$. For a member-serial group, the leading resource accounting is

$$M_{\text{live}} \approx M_{\text{fixed}} + M_{\text{prompt}}(P) + M_{\text{grad}} + \max_i M_{\text{branch}}(R_i) + M_{\text{score}}\left(\sum_i R_i\right), \quad (9)$$

$$T_{\text{update}} = T_{\text{prompt}}(P) + \sum_{i=1}^G T_{\text{score+replay}}(R_i).$$

Equation 9 bounds the live policy graph by the largest member rather than by the number of members. It does not make total memory constant in G : input/label objects, rewards, reports, and frozen old/reference scores remain $O(G)$ or $O(\sum_i R_i)$. The runners accept a configured list of group members rather than hard-coding $G = 2$ or $G = 8$. Only those two settings have Qwen receipts; larger groups remain unmeasured and

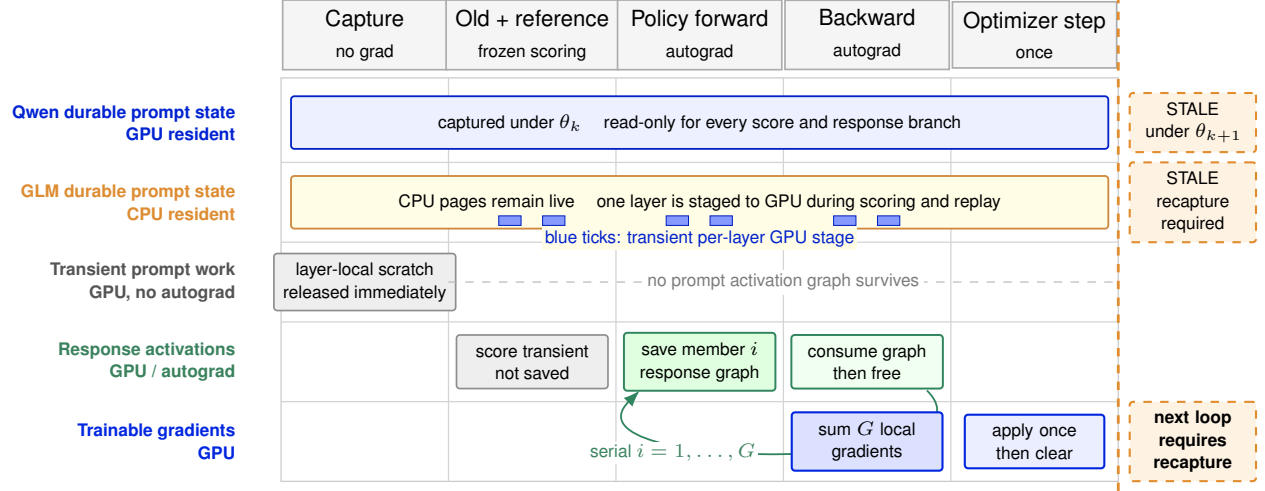


Figure 4 State lifetime across one grouped update. Durable Qwen state remains GPU-resident; GLM pages remain CPU-resident and only one layer is staged at a time. Prompt activations are released during no-grad capture, whereas only the current response graph is live under autograd. Group-indexed inputs and frozen scores may still grow with G . Worker-local gradients accumulate over G serial backwards and each optimizer is called once. The optimizer changes θ_k to θ_{k+1} , making every captured prompt state stale; a repeated loop would require recapture.

must fit the growing buffers and wall-time budget. The $G = 1$ run is only an execution canary because it cannot form a nondegenerate GRPO advantage group.

A coherent distributed update needs an additional invariant: every sharded gradient contribution must reach the parameter owner, and replicated adapters must agree after the optimizer call. A terminal worker event does not establish that invariant. The Qwen audit finds a concrete missing K/V adapter reduction; the GLM custom resident path bypasses Megatron gradient finalization, leaving CP-replicated non-expert adapter gradients unreduced before the distributed optimizer step.

4.2 State Inventory and Ownership

Table 2 distinguishes durable prompt state from transient prompt work. The distinction is the basis for both memory accounting and correctness during layerwise response replay.

Logical sharding is insufficient when a retained tensor is a view into a larger allocation. The allocator cannot release the parent while any view is alive. The Qwen implementation therefore copies every retained page shard into a right-sized physical allocation. GLM applies the same rule on CPU: a restored layer’s page table, local positions, latent pages, and indexer pages must all describe one local shard.

Stored state is immutable within one grouped update. Response branches may append suffix pages or build temporary IndexShare selections, but cannot mutate the shared prefix. Branch-local state is released after scoring or backward so every response observes identical conditioning state.

4.3 Device Placement and Layer Staging

Qwen retains compact GDN and KV state on GPU because eight-way context parallelism makes the per-rank state fit and the response attention merge reads all shards repeatedly. GLM retains its CP-local MLA and indexer-key pages on CPU. During response replay, it stages the pages needed by one layer, executes the short response layer, and releases or returns the staged copy before advancing. A shared RoPE cache avoids rebuilding position tensors for every layer and branch.

CPU placement trades transfer time for bounded residency. The trade is useful only because replay is layer-wise. Copying the complete 78-layer prefix back to GPU would recreate the storage peak. Conversely, staging tensors without preserving their page order and global positions would change the attention operator. Device placement and logical ownership are therefore one contract.

Table 2 Prompt-boundary state inventory. Durable rows survive no-grad prefix capture and are read by response replay; transient rows are released after capture and recomputed only for the short response under autograd. All replay uses a detached prompt state.

State	Placement and ownership	Scaling with prompt length P	Use during response replay
Qwen GDN recurrent state	GPU; retained per GDN layer on each CP8 rank	Fixed-size recurrent boundary state; does not grow linearly with P	Restores the recurrent state for the response transition; prompt-state gradients remain detached
Qwen full-attention KV pages	Compact GPU pages physically owned across CP8 for 16 layers	$O(P/8)$ KV storage per rank	Each response query reads every shard through the global CP8 LSE/output merge
GLM MLA latent pages	CPU; Megatron-zigzag CP32 shards for all 78 layers	$O(P/32)$ per rank; at $P = 2,097,152$, each rank owns 1,024 pages and 65,536 tokens	Materialized layer by layer as the rank-local absorbed-MLA key/value source
GLM DSA indexer-key pages	CPU; CP32 shards retained only for 21 index-computing layers	$O(P/32)$ at those 21 layers	Supplies rank-local index scores and top-2048 selection; 57 IndexShare layers consume a reused selection
GLM page and position metadata	Host page IDs and valid-token counts; replay constructs device-side positions	$O(P/(32 \times 64))$ page records at page size 64	Restores Megatron page order, global token positions, and causal bounds for local replay
Qwen transient response work	GPU attention scratch, dense-FFN activations, and temporary response pages	Prompt work is not retained; autograd storage follows the response block	Recomputed blockwise for policy backward, then released
GLM transient attention/MoE work	GPU DSA scores/top-k holder, attention scratch, router decisions, permuted rows, and selected-expert intermediates	Prompt tensors exist during no-grad capture but do not survive it; replay storage follows response length and routing	Recomputed inside whole-layer checkpointing; IndexShare state is per forward and EP32 dispatches selected response rows

4.4 Whole-Layer Checkpointing

Activation checkpointing trades retained tensors for recomputation, either at a whole block or at selected operations within it (Chen et al., 2016; Korthikanti et al., 2022). For our native MoE replay, however, the required boundary is the complete layer. Attention-only checkpointing does not bound a GLM response graph if the MoE tail retains router outputs, dispatch permutations, expert inputs, LoRA intermediates, and concatenation buffers. The working path checkpoints the complete decoder layer. Forward saves the short layer input and minimal metadata; backward recomputes attention projection, sparse selection, IndexShare publication or consumption, output projection, router decisions, expert dispatch/combine, and the selected dense or expert LoRA paths without retaining the complete layer graph across backward or any prompt activation. Checkpointing is applied to the short response graph, not the full 2,097,152-position prompt. The prompt was already evaluated without autograd. This distinction explains why checkpointing succeeds here while conventional full-sequence checkpointing still exposes large routed-token and attention workspaces during backward recomputation through the complete decoder stack.

4.5 Parallel Layout

The parallel dimensions distribute different objects. Table 3 lists the layouts used by the acceptance runs.

Qwen uses CP8 to distribute full-attention pages and recurrent state over eight GPUs. Its dense FFNs do not require expert dispatch. GLM uses TP1/CP32/EP32/ETP1/PP1. CP32 distributes long attention state. EP32 distributes 256 routed experts, nominally eight per rank. The same 32 ranks participate in both groups, but CP collectives cannot replace EP collectives: response attention needs a cross-context selection or reduction, while MoE needs token dispatch and combine across expert owners.

4.6 Adapter and Optimizer Scope

The final GLM run uses rank-8 LoRA over the configured attention projections, dense FFNs, routed and shared expert FFNs, and the output head. Base weights, embeddings, normalization parameters, router parameters, and the DSA indexer remain frozen. The Qwen path uses NF4 QLoRA over its configured dense model targets. These are parameter-efficient adapter updates; neither receipt provides evidence of

Table 3 Parallel layouts and their semantic boundaries. Context parallelism partitions prompt state, whereas expert parallelism partitions routed experts; neither layout by itself guarantees a synchronized distributed update.

Path and topology	State or expert ownership	Response-time collectives	Audited semantic and update boundary
Qwen3.6-27B 8 H20, CP8	Compact full-attention KV pages and GDN boundary state are distributed across eight context ranks; dense FFNs have no expert ownership	Forward response attention uses the global CP8 max/normalizer/value-sum LSE/output merge. The audited backward performs a dQ all-reduce only	The global response-attention forward operator is preserved, but the prompt is detached. dK/dV and adapter-gradient synchronization are missing, and AdamW is called per rank; a consistent global adapter update is therefore not established
GLM-5.2 32 H20 TP1, CP32, EP32, ETP1, PP1	CP32 owns 1,024 CPU prefix pages (65,536 tokens) per rank. EP32 nominally owns eight of 256 routed experts per rank; ETP1 and PP1 add no within-expert or pipeline split	EP32 dispatches and combines routed response rows. In the accepted receipt, DSA scoring, top-2048 selection, and sparse attention remain local to each CP shard; no cross-CP candidate or attention-output merge executes	All ranks report two 78-layer backwards and a terminal DistOpt call. The historical custom path skips <code>finalize_model_grads</code> , so CP-replicated non-expert adapter gradients are unreduced. The prompt is detached and the response objective is CP-local

full-parameter model training (Hu et al., 2021; Dettmers et al., 2023). Rank allocation, adapter sharing, mixed-precision fine-tuning, and pruning are complementary design axes (Zhou et al., 2025a,e, 2026a,b, 2025b,c); LongStraw keeps those choices fixed and targets the execution boundary and distributed-state contract examined throughout the audited replay path.

All old-policy values are frozen before policy replay. In the fresh GLM acceptance run, the old snapshot is also the reference for the first update. With $\epsilon = 0.2$ and $\beta = 0.01$, the initial importance ratios are one and measured KL is zero. The run exercises the recorded branch and optimizer-call ordering, but it does not exercise an active clip boundary or a nonzero first-step KL penalty.

5 Qwen: Dense Hybrid Replay within an Eight-H20 Budget

The Qwen path is the cleaner of the two architecture cases because it has no expert router, no expert-parallel token exchange, and no sparse index whose meaning changes under context sharding. It is nevertheless not a conventional data-parallel training job. The implementation replicates the dense model and response query computation across eight ranks while distributing the long key/value sequence of each full-attention layer. This section opens that contract at the tensor level. It documents a verified global conditional forward operator inside the fixed eight-H20 execution envelope; model-parallel gradient composition remains open.

5.1 Hybrid Layer Anatomy and the Prompt Boundary

The audited model snapshot contains 64 decoder layers at hidden width 5,120, with a repeating pattern of three `linear_attention` entries and one `full_attention` entry (Qwen Team, 2026). The audited runtime maps the former to its gated-delta-network (GDN) module, giving 48 GDN layers and 16 full-attention layers. The recurrent mechanism follows Gated DeltaNet (Yang et al., 2025). In the pinned snapshot, each full-attention layer has 24 query heads, four KV heads, head dimension 256, an output gate, and a partial rotary factor of 0.25 (Qwen Team, 2026). The 24-to-4 query/KV arrangement follows grouped-query attention (Ainslie et al., 2023), while its rotary position mechanism follows RoPE (Su et al., 2021). Every layer uses the same dense gated FFN shape, with intermediate width 17,408. Ignoring biases, one FFN therefore contains

$$3 \times 5,120 \times 17,408 = 267,386,880 \quad (10)$$

base weights. Every token activates all three projections; there is no conditional expert capacity to distribute.

The two token mixers leave different state at the end of a no-grad prompt. For a GDN layer, the boundary consists of a recurrent matrix and the final three-token convolution tail. Its shape is independent of prompt length. For a full-attention layer, the boundary is every prompt key and value, so storage is linear in prompt length. Prompt capture evaluates the complete dense stack, including every FFN, but releases prompt hidden

states, FFN intermediates, and temporary mixer work after each chunk. Only 48 compact GDN boundaries and 16 full-attention KV page sets survive. Thus pages serve the 16 quadratic-history layers, while recurrence removes length-dependent state from the remaining 48 GDN layers.

The NF4 QLoRA path uses rank 16, scaling 32, learning rate 2×10^{-4} , and zero weight decay. LoRA targets full-attention, GDN, and dense-FFN projections, giving 116,727,808 trainable parameters. NF4 reduces persistent base-weight storage, but it does not reduce KV length or response activation lifetime (Dettmers et al., 2023). Those two terms are handled by physical context parallelism and blockwise replay.

5.2 From Logical Shards to Physical Pages

Let page size be $S = 64$, global page index be p , and the CP world size be $C = 8$. Page ownership is block-cyclic,

$$\text{owner}(p) = p \bmod C. \quad (11)$$

Each rank retains only its owned pages but keeps the global logical KV length. Thus RoPE positions, query start indices, and causal masks use original sequence coordinates, not compacted local ones.

This distinction was initially only logical. A page obtained by slicing a 4,096-token prefix chunk could be contiguous as a tensor while still retaining the storage of the complete parent chunk. Discarding the unowned slice did not release that parent allocation. The manager instead allocates right-sized page tensors and copies owned slices. PagedAttention manages serving KV through logical-to-physical block tables over fixed-size physical blocks (Kwon et al., 2023). At our training boundary, copying owned slices makes allocator ownership agree with page-table ownership, so releasing a logical page also releases its physical storage.

The arithmetic beyond two million context positions is explicit. The reported sequence has 2,088,960 prompt tokens followed by 8,192 response-input tokens, for an exact context length of 2,097,152. The prompt contains

$$N_{\text{page}} = \frac{2,088,960}{64} = 32,640, \quad N_{\text{page/rank}} = \frac{32,640}{8} = 4,080. \quad (12)$$

For BF16 K and V with four KV heads and head dimension 256, the raw prompt KV payload retained by one rank across the 16 full-attention layers is

$$B_{\text{KV/rank}} = 16 \times 4,080 \times 2_{\{K,V\}} \times 64 \times 4 \times 256 \times 2 \text{ bytes} \quad (13)$$

$$= 17,112,760,320 \text{ bytes} \approx 15.94 \text{ GiB}. \quad (14)$$

The measured allocator footprint is larger because it includes quantized weights, adapters, recurrent state, page metadata, response pages, and temporary kernels. Equation 14 exposes the ownership slope: each additional prompt page resides on one CP rank per full-attention layer, not on all eight ranks.

5.3 Global Full-Attention Forward Composition

For response query t , let $\mathcal{K}_r$ be the keys owned by rank r and let $s_{t,j} = q_t^\top k_j / \sqrt{d}$. A local paged-attention kernel returns a normalized local output and its row log-normalizer,

$$\ell_{r,t} = \log \sum_{j \in \mathcal{K}_r} e^{s_{t,j}}, \quad o_{r,t} = \frac{\sum_{j \in \mathcal{K}_r} e^{s_{t,j}} v_j}{e^{\ell_{r,t}}}. \quad (15)$$

The global output is reconstructed without moving the KV pages.

$$m_t = \max_r \ell_{r,t}, \quad (16)$$

$$a_t = \sum_r e^{\ell_{r,t} - m_t}, \quad (17)$$

$$n_t = \sum_r e^{\ell_{r,t} - m_t} o_{r,t}. \quad (18)$$

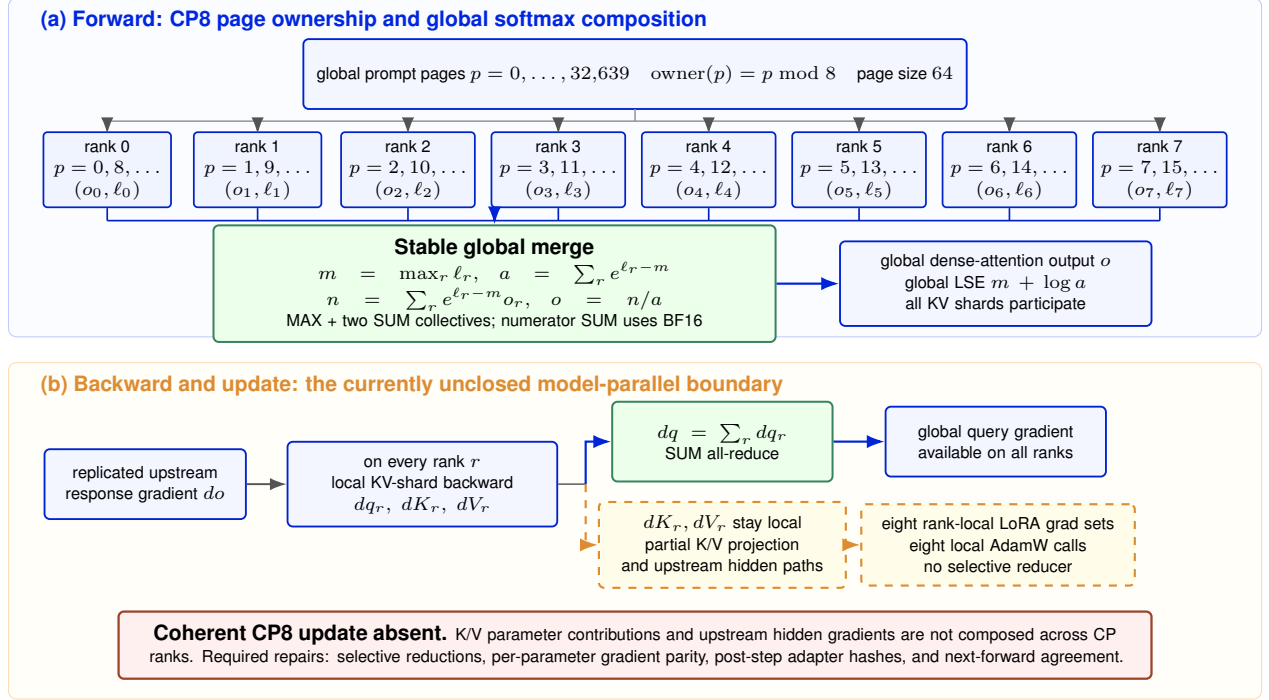


Figure 5 Qwen CP8 forward fidelity and backward synchronization gap. The full-attention forward partitions physical KV pages and recomposes the global softmax over all shards, subject to the BF16 numerator reduction. Backward all-reduces the query gradient, but K/V-derived parameter and hidden-gradient contributions remain rank local before eight local optimizer calls. The receipt therefore establishes global conditional-forward and update-shaped execution capacity, not a coherent distributed model update.

Then

$$o_t = \frac{n_t}{a_t}, \quad \ell_t = m_t + \log a_t. \quad (19)$$

This requires one MAX all-reduce and two SUM all-reduces per composed output. It is the same stable log-sum-exp identity used by exact tiled attention (Rabe and Staats, 2021; Dao et al., 2022). It reconstructs the dense conditional response-attention operator from disjoint KV partitions.

Production reduces n_t in BF16 to lower communication payload. The maximum, denominator, and global LSE remain FP32. Here “exact” means exact partition composition of the specified dense operator under that finite-precision reduction, not bitwise equality with unsharded FP32 execution.

5.4 Four-Block Response Replay

The shared prompt is captured once in 510 chunks of 4,096 tokens. A response branch is then split into four 2,048-token blocks. The branch first runs a no-grad suffix pass. At each block boundary it clones the input GDN states and appends that block’s KV pages. The policy pass traverses the four blocks in reverse. For one block it restores the corresponding GDN input state with gradient tracking, recomputes all 64 decoder layers under whole-layer checkpointing, forms causal selected-token log-probabilities, runs backward, propagates the recurrent/conv-state gradient to the preceding block, and pops the temporary KV update. Only one 2,048-token response graph is live at a time, so block count increases replay work without retaining all block graphs together across the complete suffix on each device.

For each group member the observed ordering is old scoring, reference scoring, policy suffix forward, and policy reverse. Members are serialized after the shared prompt. Their parameter gradients accumulate until one optimizer call is issued after the last member. This schedule explains why, for this workload, increasing group size mainly increases time rather than peak memory.

There are three workload qualifications. First, the inputs are synthetic random tokens and rewards are deterministic functions of group index. Second, the old and reference scores are produced by the same current model before the step; the run uses $\beta = 0$, so it does not exercise an independent reference policy or an active KL term. The live policy expression is the unclipped ratio term. Because old and current scores coincide at the first step, the ratio is one and clipping would be inactive even if present. Third, 8,192 is the number of response-input tokens. The causal selector scores labels from response index one onward, yielding at most 8,191 suffix targets per branch under the audited implementation. The terminal logs record the response length but do not serialize this scored-token count.

5.5 The Fixed Eight-H20 Execution Envelope

Both runs use the fixed eight-NVIDIA-H20 budget, CP8, page size 64, a 4,096-token prefix chunk, a 2,048-token response block, and 512-token FFN microblocks during backward. CUDA peak counters are reset after model, adapter, optimizer, and input construction. Reported memory is decimal GB from `max_memory_allocated`, not reserved memory or process memory from `nvidia-smi`. Timings exclude model loading.

Group size two. The reported rank records 5,198.780 seconds end to end and a 97.503 GB peak. The shared prefix consumes 4,656.225 seconds, about 89.6% of the reported time. Across the eight terminal records, total time ranges from 5,196.750 to 5,200.684 seconds; local optimizer calls take 0.155–0.200 seconds.

Group size eight. The reported run completes in 6,785.225 seconds at a 97.711 GB peak. Prefix capture takes 4,653.420 seconds. Group zero completes at 4,931.993 seconds; each of the seven additional serialized members adds a median 264.739 seconds. After the first member, a typical branch spends about 32.0 seconds in old scoring, 32.0 seconds in reference scoring, 31.2 seconds in suffix forward, and 169.1–169.6 seconds in reverse replay. Terminal times across ranks range from 6,783.943 to 6,785.225 seconds, while the corresponding local optimizer-call timers range from 0.163 to 0.186 seconds.

The increase from $G = 2$ to $G = 8$ is therefore 1,586.445 seconds but only 0.208 GB (0.213%) at the reported peak. Removing the shared prefix, the post-prefix cost per member is 271.278 versus 266.476 seconds. Amortizing the prefix reduces mean wall time per supplied response from 2,599.390 to 848.153 seconds. These derived values come from two single runs, not a fitted scaling law. They establish that the requested scoring, four-block policy backwards, gradient materialization, and terminal optimizer calls fit at a context length of 2,097,152. They do not establish throughput-optimal group parallelism, online rollout, policy improvement, or repeated fresh-prefix training.

5.6 Why the Receipt Is Not Yet a Coherent CP8 Update

The forward collective is complete, but model-parallel backward is not. Each attention rank computes a local query-gradient contribution dq_r and local gradients for its K/V tokens. The correct query gradient is

$$dq = \sum_{r=0}^7 dq_r, \quad (20)$$

and the custom backward performs this all-reduce. In contrast, its dK_r and dV_r remain rank local. That is valid for sharded KV storage, but the K/V projection LoRA weights are replicated. Their correct parameter contributions require composition across the disjoint token owners,

$$\nabla W_K = \sum_r \nabla W_K^{(r)}, \quad \nabla W_V = \sum_r \nabla W_V^{(r)}, \quad (21)$$

with K/V hidden-state contributions composed before differentiating earlier replicated layers.

The audited runner initializes only NCCL: it has no DDP wrapper, parameter-gradient reducer, or selective model-parallel composition required by Equation 21. Each rank owns an AdamW instance (Loshchilov and Hutter, 2019) and steps locally. All-reducing every completed gradient would still be wrong: query contributions are already global but K/V contributions are not, so the two paths require reductions at their distinct parameter-ownership boundaries for replicated adapters.

No terminal receipt records a gradient norm, parameter delta, post-step adapter hash, or next-forward replica comparison. The supported statement is therefore narrower than “a coherent distributed update”: the implementation executes global full-attention response forwards, response-shaped backward graphs, and eight terminal optimizer calls within the fixed eight-H20 envelope at 2,097,152 context positions. We call this an *execution/update-shaped capacity receipt*. Closing the Qwen path requires model-parallel K/V and hidden-gradient composition, followed by per-parameter gradient parity and post-step replica-consistency tests. The detached prompt-state term in Equation 3 remains a separate limitation even after that synchronization is repaired and numerically validated against a tractable reference.

6 GLM: Paged MLA/DSA Replay within a 32-H20 MoE Budget

The GLM path is architecturally more demanding. Its long-range token mixer is neither dense attention nor a prompt-length-independent recurrence. Each decoder layer combines multi-head latent attention (MLA) with a dynamic sparse-attention (DSA) index, while most feed-forward blocks are routed MoE layers. Consequently, a useful prompt boundary must preserve two attention representations, reproduce the cross-layer IndexShare schedule, and re-enter the expert-parallel tail under autograd. Traversing 78 layers is therefore necessary but does not prove a faithful global full-context operator or coherent distributed update; we state the implemented path and missing tensor-level collectives separately.

6.1 Layer Anatomy: MLA/DSA Before Dense or Routed FFNs

The audited configuration has hidden width $H = 6,144$ and 78 decoder layers. Attention uses 64 MLA query heads, query and KV latent ranks 2,048 and 512, and a 64-dimensional positional channel. The absorbed key retained by the runtime therefore has width $512 + 64 = 576$. The DSA indexer uses 32 heads of dimension 128 and selects 2,048 key positions per response query. These dimensions follow the released GLM architecture and the inspected runtime configuration artifact (zai-org, 2026).

Not every layer recomputes the index. With index frequency four and skip offset three, the zero-based index-computing set is

$$\mathcal{I}_{\text{compute}} = \{0, 1, 2\} \cup \{6, 10, 14, \dots, 74\}, \quad |\mathcal{I}_{\text{compute}}| = 21. \quad (22)$$

The other 57 layers are IndexShare consumers. A compute layer publishes its top- k tensor to a carrier scoped to one decoder forward; consumers reuse it instead of retaining prompt-length index state. This implements cross-layer index reuse, not a per-layer cache (Bai et al., 2026).

The FFN schedule is similarly nonuniform. Layers 0–2 are dense gated FFNs. The remaining 75 layers have 256 routed experts, top-8 routing, and one shared expert. With intermediate width 2,048, one routed expert contains, ignoring biases,

$$B_{\text{expert}} = 3 \times 6,144 \times 2,048 = 37,748,736 \quad (23)$$

base weights. A sparse layer contains approximately 9.66 billion routed expert weights, but one token evaluates eight experts, or about 302 million routed weights, plus the shared branch. MoE reduces activated parameter count; it does not remove router, permutation, all-to-all, expert-input, and combine tensors from the layer execution (Lepikhin et al., 2021; Fedus et al., 2022; DeepSeek-AI, 2024a,b).

The final run uses rank-8 LoRA on eight target categories: the query and KV down/up projections, the attention output projection, both FFN projections, and the output head. The FFN patterns match dense, routed-expert, and shared-expert modules. Base weights, embeddings, normalization parameters, DSA indexer projections, and router parameters remain frozen. The router’s `expert_bias` is a non-parameter state buffer; its transition is not recorded in the final receipt.

6.2 TP1/CP32/EP32 Assigns Two Different Kinds of Ownership

The acceptance topology is TP1/CP32/EP32/ETP1/PP1 on 32 H20 GPUs. TP1 leaves each attention and dense projection structurally intact. CP32 distributes the prompt token axis and therefore the retained MLA/DSA state. EP32 distributes the 256 routed experts, nominally eight experts per rank. CP and EP

contain the same workers in this run, but they are not interchangeable dimensions: CP answers which rank owns a prompt position, whereas EP answers which rank owns an expert parameter shard (Liu et al., 2025).

The distinction appears directly in the data movement. During prefix capture, one rank processes 65,536 prompt tokens. Top-8 routing expands that shard to $65,536 \times 8 = 524,288$ routed rows before load skew. A single balanced BF16 hidden buffer of shape $[524,288, 6,144]$ occupies

$$524,288 \times 6,144 \times 2 = 6,442,450,944 \text{ bytes} = 6 \text{ GiB.} \quad (24)$$

This is a derived lower-level buffer size, not a measured whole-layer peak. The live MoE path also needs dispatch metadata, output storage, inverse permutations, shared-expert work, and LoRA intermediates. They are released during capture because autograd is disabled; a conventional beyond-2M graph cannot. Successive full-sequence runs therefore moved from DSA scratch OOMs to expert-LoRA and expert-output OOMs.

6.3 Zigzag Context Pages within the 32-H20 Route

Let the prompt length be $P = 2,097,152$, the physical page size be $S = 64$, and the CP size be $C = 32$. Megatron’s context-parallel layout first partitions the global sequence into $2C = 64$ contiguous chunks. Each chunk contains

$$N_{\text{page}} = \frac{P}{S} = 32,768 \text{ pages}, \quad N_{\text{chunk}} = \frac{N_{\text{page}}}{2C} = 512 \text{ pages.} \quad (25)$$

Rank $r \in \{0, \dots, 31\}$ owns chunk r and its mirrored chunk $63 - r$. Its ordered global page set is therefore

$$\mathcal{P}_r = \{512r, \dots, 512r + 511\} \parallel \{512(63 - r), \dots, 512(63 - r) + 511\}, \quad (26)$$

where $\parallel$ denotes concatenation in the local tensor order. Every rank stores 1,024 pages, or 65,536 tokens. For example, rank 0 stores pages 0–511 followed by 32,256–32,767; rank 31 stores 15,872–16,383 followed by 16,384–16,895. The final log records all 32 page samples and agrees with this formula.

The page manager copies into right-sized CPU pages, not views of larger GPU capture tensors. PagedAttention manages serving KV through fixed-size physical blocks and a logical-to-physical block table (Kwon et al., 2023). In our training page manager, memory is bounded only when allocator ownership matches page-table ownership. Global page IDs stay attached to local tensor order, so DSA masks and selections retain their original global coordinates throughout capture and replay.

6.4 Stored Tensor Contract and Derived Residency

Every layer stores one absorbed MLA page component. Its runtime key is `m1a_latent_kv_pages`. The 21 index-computing layers additionally store DSA index-key pages under `dsa_indexer_key_pages`; an IndexShare consumer stores no duplicate index key. Earlier live shape diagnostics record both components as BF16 and the final page contract follows from those inspected tensors:

$$\text{MLA page : } [1, 64, 1, 576], \quad \text{local materialization : } [1, 65,536, 1, 576], \quad (27)$$

$$\text{DSA key page : } [1, 64, 128], \quad \text{local materialization : } [1, 65,536, 128]. \quad (28)$$

The final manifest records component names, page counts, and CPU placement, but omits component dtype and shape. The shapes above are contract-derived and cross-checked against the earlier live trace rather than measurements serialized directly in the rank-complete final manifest.

The corresponding retained-state arithmetic is exact once that contract is assumed. Per rank, one layer’s MLA pages occupy

$$1,024 \times 64 \times 576 \times 2 = 72 \text{ MiB}, \quad (29)$$

and one compute layer’s index pages occupy

$$1,024 \times 64 \times 128 \times 2 = 16 \text{ MiB.} \quad (30)$$

Thus the CPU-resident prompt state is

$$B_{\text{CPU/rank}} = 78 \times 72 \text{ MiB} + 21 \times 16 \text{ MiB} \quad (31)$$

$$= 5,952 \text{ MiB} = 5.8125 \text{ GiB}, \quad (32)$$

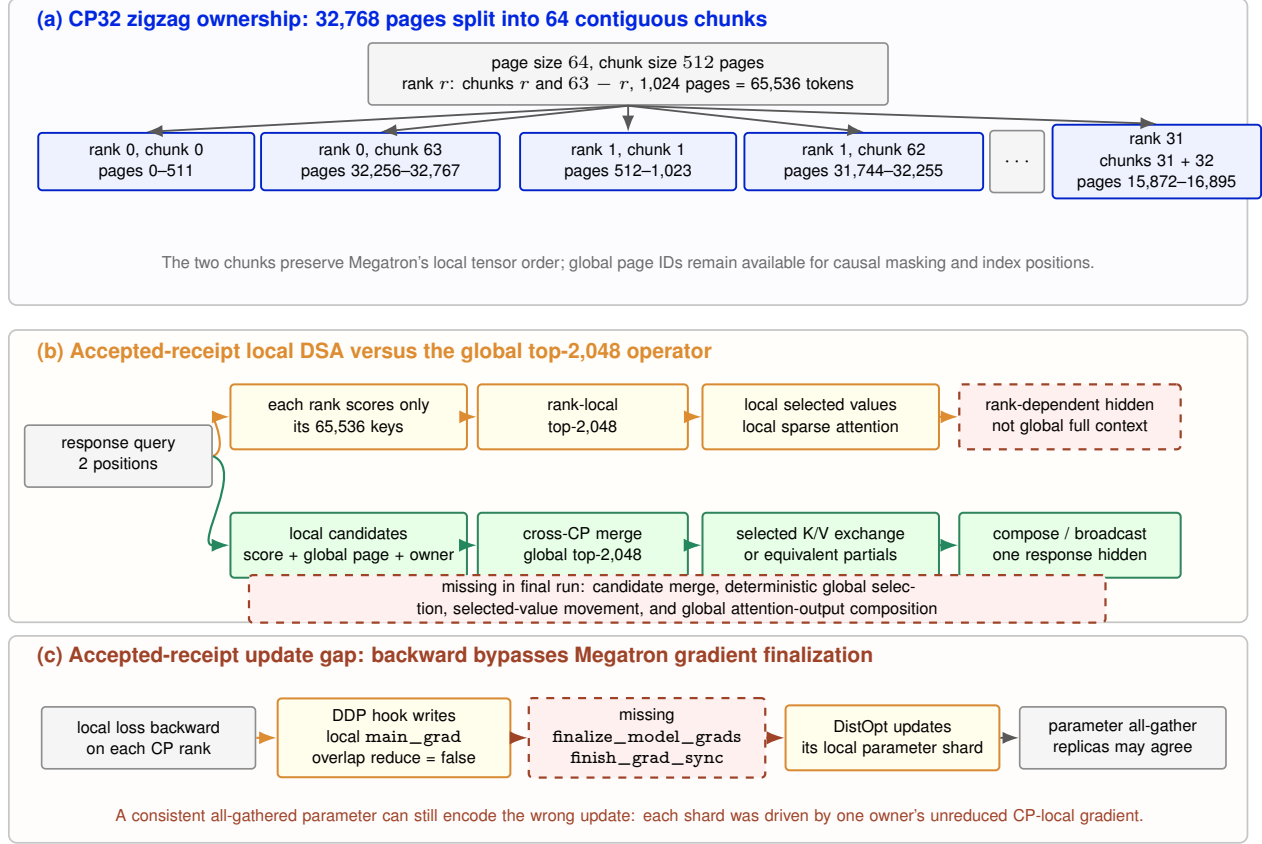


Figure 6 GLM CP32 ownership and the two missing global operations. Megatron zigzag partitioning gives each rank two mirrored 512-page chunks. The accepted-receipt DSA fallback selects top-2,048 independently inside each local 65,536-token shard. A faithful full-context operator needs a cross-rank candidate merge, selected-value exchange, and one composed response hidden state. Separately, the accepted-receipt backward writes local DDP gradient buffers and calls the distributed optimizer without Megatron’s gradient-finalization reduction. Parameter all-gather may restore replica equality, but it cannot reconstruct the omitted CP gradient sum.

or 186 GiB across 32 ranks. Layerwise staging uses 72 MiB for an IndexShare layer and 88 MiB for an index-computing layer. These payloads exclude response activations and kernel workspaces, so they are not whole-step peaks. The final log reads the retained CUDA peak after prefix capture, before policy replay.

6.5 One Replayed Layer: State, Index, Attention, and MoE

The response path consumes two scored positions; its decoder input has shape $[2, 1, 6, 144]$. The runner materializes that rank’s 65,536 prompt positions, recomputes the response-side projections, and forms a 65,538-position local attention problem. An index-computing layer evaluates the response queries against its local DSA key and publishes indices with shape $[1, 2, 2, 048]$. An IndexShare layer consumes the previously published indices. Both paths run the runtime’s unfused absorbed sparse attention, the live output projection and bias/dropout/add path, and the dense-or-MoE `_forward_mlp` tail.

For a routed layer, each local response row is expanded to eight assignments, permuted by expert owner, exchanged through the EP all-to-all, evaluated by the owner’s expert FC1/FC2 LoRA path, and inverse-combined with router probabilities. The shared expert branch is evaluated in parallel and added to the routed result. Thus the final trace is not an attention-only or synthetic FFN surrogate: all 75 MoE tails execute under the native EP autograd graph. What the trace does not retain is a final-run router histogram, per-expert row count, all-to-all payload, or communication timing.

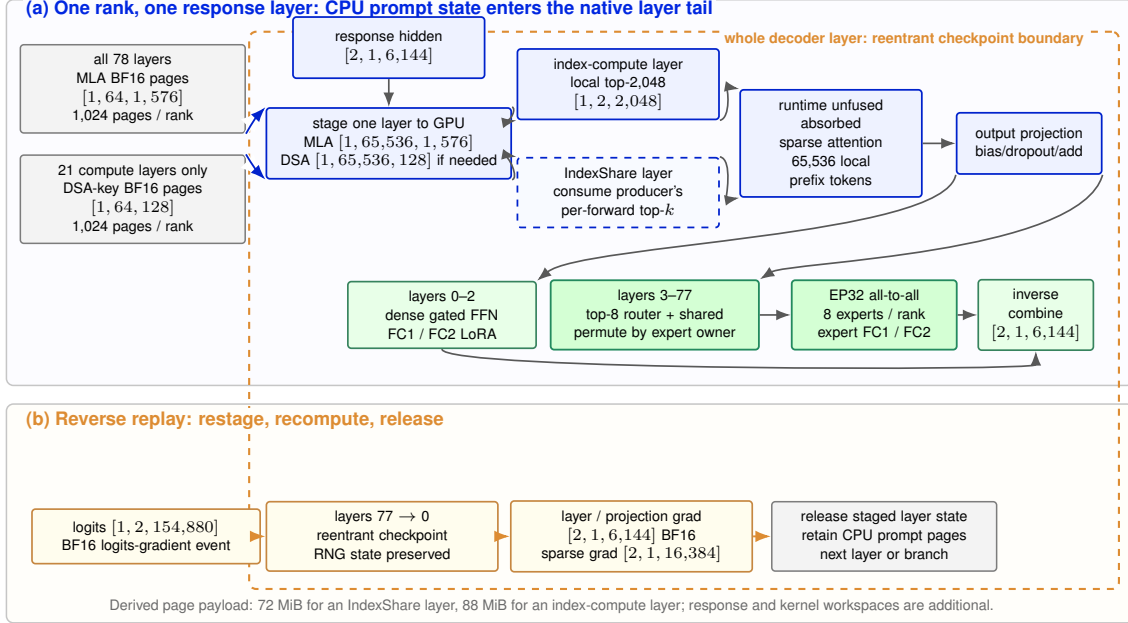


Figure 7 GLM resident layer replay and whole-layer checkpointing. CPU pages retain only the CP-local prompt state. A compute layer stages MLA and DSA-key pages and publishes a local top-2,048 tensor; an IndexShare layer stages MLA pages and consumes the matching per-forward selection. The live attention projection and dense or EP32 MoE tail execute inside one checkpoint boundary. Backward traverses layers in reverse, restages and recomputes one layer, emits the audited tensor shapes, and releases its workspace. Page shapes and byte counts are contract-derived; they are not whole-step peak measurements.

6.6 Whole-Layer Checkpointing and the Recorded Backward

Checkpoint placement controls which intermediates survive and which operators are recomputed (Chen et al., 2016; Korthikanti et al., 2022). Attention-only checkpointing left the routed tail live, so final policy checkpoints whole decoder layers reentrantly with RNG preservation. Forward retains only the short $[2, 1, 6, 144]$ input and metadata. Reverse re-enters layers 77 through 0, restages CPU pages, recomputes sparse attention plus the dense/MoE tail, and releases workspace; saved sparse-attention tensors may be offloaded to CPU.

The trace shapes make this lifecycle observable. A representative policy trace records:

- decoder input and every layer output: $[2, 1, 6, 144]$ BF16;
- local top- k : $[1, 2, 2, 048]$, with 65,536 prefix positions;
- logits: $[1, 2, 154, 880]$, followed by a BF16 logits-gradient event;
- layer and attention-projection gradients: $[2, 1, 6, 144]$ BF16;
- sparse-attention output gradients: $[2, 1, 16, 384]$ BF16.

Policy and old-policy traces contain 396 and 160 events, respectively. Two members, two phases, and 32 ranks produce 128 files with 35,584 JSONL events. Each policy rank records 78 checkpointed layer ends and 78 each of layer, attention-projection, and sparse-attention backwards. This strongly shows the architecture-shaped path completed twice, but contains no parameter-gradient tensors or cross-rank reductions.

6.7 Why the DSA Operator in the Accepted Receipt Is Local, Not Full Context

For response query q_t , rank r in the accepted receipt computes index scores only against its page set $\mathcal{P}_r$, then selects

$$I_{r,t} = \text{TopK}_{j \in \mathcal{P}_r} \left[\sum_{h=1}^{32} w_{t,h} \phi(q_{t,h}^\top k_j) \right], \quad |I_{r,t}| = 2,048, \quad (33)$$

with causal masking in global page coordinates. Selected values yield rank-local hidden states; CP ranks merge neither candidates nor outputs. Disjoint prompt ownership makes $I_{r,t}$ rank-specific. Each layer therefore realizes local top-2,048 over 65,536 tokens, not model-global top-2,048 over the full 2,097,152-position prompt.

A faithful distributed operator first needs sufficient local candidates with global scores, positions, and owner IDs; a deterministic cross-rank merge must then compute

$$I_t = \text{TopK}_{j \in \bigcup_r \mathcal{P}_r} s_{t,j}, \quad |I_t| = 2,048. \quad (34)$$

The selected K/V rows must be exchanged from their owners, or their attention numerators and normalizers must be composed with an equivalent distributed operator. Finally, all ranks that continue the replicated decoder must agree on the composed response hidden state. Candidate merge, selected-value exchange, and output composition are all absent from the final run. This is the primary forward-fidelity gap.

6.8 Gradient Finalization Audit: A Terminal Call Is Not an Update

The optimizer audit exposes a second, independent gap. LoRA transformation is applied before Megatron DDP construction, so trainable adapter parameters are registered in DDP gradient buffers. The inspected DDP configuration has `overlap_grad_reduce=false`. Its parameter hooks therefore add each local autograd gradient into `param.main_grad`, but they do not launch an all-reduce or reduce-scatter during backward. In the conventional Megatron schedule, the post-backward `finalize_model_grads` call invokes `finish_grad_sync` on every model chunk. This is the required DP×CP gradient reduction for non-expert replicas.

The resident path does not call that schedule. It runs two local `loss.backward()` calls and then invokes `engine.optimizer_step()` directly. The concrete engine method calls the distributed optimizer’s `step()` without first finalizing model gradients. The distributed optimizer then copies the shard it owns from the unreduced full gradient buffer, updates that parameter shard, and all-gathers the updated parameter shards. In symbols, the required non-expert gradient is

$$g^* = \sum_{r=0}^{31} g_r, \quad \text{but the observed path supplies shard owner } r(s) \text{ with } g_{r(s)}|_s. \quad (35)$$

The subsequent parameter all-gather can make complete parameter replicas look mutually consistent even though different parameter shards were updated from different local objectives. Parameter equality after all-gather would not repair the missing gradient sum. In this run the local gradients are expected to differ because each rank’s DSA forward sees a different 65,536-token shard.

The affected class is the CP-replicated, non-expert LoRA state: attention projections, the three dense FFNs, the output head, and any other adapter placed in the non-expert DP×CP buffer. Routed experts require a different interpretation. With world size 32, EP32, ETP1, and PP1, expert data-parallel size is one. Routed-expert adapter weights are marked `allreduce=false`, are uniquely owned, and receive token contributions through the differentiable EP dispatch/combine path; no replica average is required for those unique expert shards. Shared experts have an additional PEFT EP gradient hook in the inspected runtime, but the final trace does not record per-module hook coverage or hashes, so their synchronization is not promoted to a run-level claim. The router `expert_bias` state is also unrecorded.

The final manifest contains no CP finalization event, per-rank parameter-gradient count or norm, optimizer-state hash, pre/post adapter hash, parameter delta, post-all-gather checksum, or next-forward equality test. Its terminal evidence must therefore be read literally: all 32 workers completed two architecture-shaped local backwards and invoked the distributed optimizer once per worker. It does *not* establish a correct CP32-reduced GRPO gradient or a coherent full-context GLM update.

Receipt classification. The strongest supported description is a *fixed-budget architecture-shaped, rank-complete resident replay/backward capacity receipt at 2,097,152 positions*. Converting it into a model-faithful distributed training result requires, in order, global DSA candidate/output composition, restoration of `finalize_model_grads`, per-parameter gradient and optimizer-delta parity against a short-context conventional reference, and post-step replica checks. Only then can online rollout, reward execution, repeated updates,

Table 4 The failure ladder from a conventional GLM full-sequence path to the fresh 2,097,152-position $G = 2$ rank-complete execution receipt. Each gate proves only the scope in the final column.

Gate	Blocking observation	Change	What passed
Full sequence	32K passes; 2.097M OOM moves from DSA scratch to expert LoRA and MoE concatenate	Stop retaining a full-context autograd graph	Establishes the need for a detached prompt boundary
Prefix capture	No durable GLM state contract	Capture MLA latent + DSA index-key pages, no grad	128K, 256K, 512K, 1M, then 2.097M prefix
Layer-0 replay	1M unchunked MoE replay OOM	Chunk/offload prefix state; recompute a short suffix	Layer-0 1M and 2.097M backward + optimizer-call canaries
All-layer replay	Missing IndexShare holder, CP guard, in-place views	Publish/reuse top-k holder; repair layer tail	Bounded 32K multi-layer gates
Topology/state	All-layer working set remains too large	TP1/CP32/EP32, CPU pages, layer checkpointing	32K all-layer, then 64K all-layer
2.097M single member	Group semantics untested	All 78 layers, response replay, one backward	$G = 1$ backward + optimizer-call canary
Fresh grouped run	Old values and local group gradients must remain ordered	Freeze both old scores, run two serial backwards, call DistOpt once/worker	$G = 2$, 32/32 ranks terminate; CP-local DSA and missing CP grad finalize

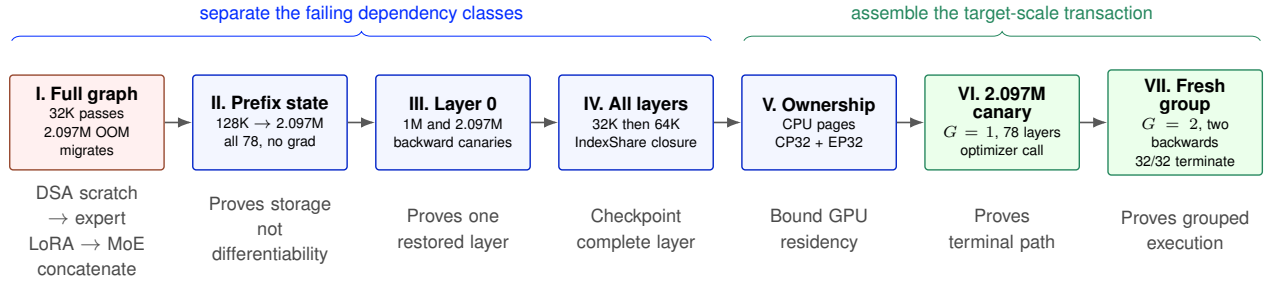


Figure 8 The staged route to a 2.097M GLM path in 32 H20s. Each gate adds one dependency class only after the narrower gate passes. Prefix capacity, differentiable replay, all-layer closure, grouped ordering, and semantic fidelity remain separate tests.

and quality evaluation answer the separate question of whether the method trains a useful policy under the stated budget and context.

7 Making the GLM GRPO Path Fit a Fixed 32-H20 Budget

The final grouped run is the endpoint of a budget-constrained progression. We held the 32-H20 allocation fixed at TP1, CP32, EP32, ETP1, and PP1 and moved one limiting resource at a time. Each gate isolated one dependency class: full-graph memory, prefix-state capacity, differentiable replay, cross-layer architecture state, device placement, or group ordering. This section reconstructs the progression as a systems argument. Table 4 summarizes the gates; Figure 8 tracks the limiting resource across graph boundaries.

7.1 Stage I: Full-Graph Bottleneck Localization

A conventional full-sequence LoRA path passed at 32K. Extending the same graph to a 2,097,152-position prompt did not reveal one dominant allocation that could be optimized in isolation. Reducing or bypassing one peak moved the OOM to the next part of the layer. The observed sequence was DSA attention scratch, expert-LoRA scale/add work, and finally MoE output concatenation near the H20 memory limit.

This movement is consistent with the GLM layer anatomy. DSA avoids dense core attention over every key, but its indexer must still process the long context. MoE evaluates a subset of experts per token, but full-sequence backward retains routing, permutations, selected expert inputs, and adapter intermediates. At a 65,536-token shard, one fully expanded routed hidden buffer can already occupy 6 GiB, as derived in Section 3. Optimizing sparse attention alone therefore cannot bring the complete beyond-2M autograd graph within the per-rank H20 memory budget for this workload.

The conclusion from this stage was structural: the long prompt could not remain inside the differentiable graph. The next tests separated prompt-state capacity from response replay.

7.2 Stage II: Prefix-State Scaling

The prefix-only path evaluated all 78 layers without autograd and retained MLA latent pages plus DSA indexer-key pages. It was scaled through 128K, 256K, 512K, 1M, and 2,097,152. The final pass established that the full decoder could process the prompt, CP-sharded pages could cover it, and prompt MoE work could be released per layer without retaining a backward graph.

Prefix capture did not establish trainability: it omitted state restoration, response logits, IndexShare lifetime, expert routing under autograd, and an optimizer event. A scale-only prefill is not a training result.

7.3 Stage III: Single-Layer Differentiable Replay

Layer-0 replay provided the first small differentiable slice. The runtime restored one layer’s prompt pages, appended a short response, produced a live loss, and ran backward. Early 32K and 128K gates checked the tensor contract. At 1M, GPU-resident state and MoE work became costly. CPU-resident pages and chunked staging then enabled layer-0 backward plus optimizer-call canaries at 1M and 2,097,152.

This stage separated two questions that are often conflated: whether the prompt state fits, and whether a response query can consume it under autograd. The layer-0 result answered both for one layer, but it could not exercise cross-layer IndexShare or accumulate activation lifetime through the 78-layer stack.

7.4 Stage IV: All-Layer Architecture Closure

Short 32K and 64K all-layer runs exposed bugs invisible to prefix-only and layer-0 tests.

IndexShare lifetime. An index-computing layer must publish a top-k selection to a per-forward holder, and dependent layers must consume the matching selection. The holder cannot be global across response branches, and checkpoint recomputation must reproduce the same producer/consumer order.

Resident DSA call contract. The stock non-packed DSA path expected query and key layouts compatible with its CP all-gather guard. A short resident query over long saved pages did not match that contract. The acceptance runner used the runtime’s unfused absorbed MLA sparse fallback over materialized local pages. This fixed execution but introduced the CP-local semantic boundary described in Section 6.

Views and in-place operations. Restored-page views tolerated in no-grad capture became invalid under autograd. Explicit ownership and mutation removal closed the layer tail.

Activation lifetime. Checkpointing only attention retained MoE routing and expert activations. A complete-layer boundary made saved activations scale with response length.

The resulting 32K and 64K gates executed all 78 attention and FFN tails, including 21 index producers, 57 IndexShare consumers, three dense FFNs, 75 MoE FFNs, and their backward paths.

7.5 Stage V: Parallel Ownership and CPU Pages

The acceptance topology uses TP1/CP32/EP32/ETP1/PP1. CP32 assigns a portion of the long prompt pages to every rank. EP32 assigns the 256 routed experts, nominally eight experts per rank. CPU page storage and one-layer staging bound attention-state residency; complete-layer checkpointing bounds response activations. A shared RoPE cache avoids repeated layer-local position allocation.

This topology is a budget choice, not a generic recipe for every MoE model. It reuses the same 32 ranks for CP and EP rather than scaling a second device group; attention state uses all ranks for CP, and the routed expert set is large enough to use those ranks for EP. Both communication patterns still execute.

7.6 Stage VI: A 2.097M Single-Member Canary under the Fixed Budget

Before introducing group accumulation, a $G = 1$ canary ran the 2,097,152-position prompt, all 78 response layers, one backward, and one optimizer call per worker under the same 32-H20 allocation. This gate checked that state restoration, checkpoint recomputation, local gradient materialization, and the terminal optimizer path could coexist within the target per-rank H20 storage envelope.

The canary was not a valid GRPO group and did not test group ordering: one member needs neither frozen old scores nor cross-response gradient accumulation. The grouped acceptance run started in a fresh process.

7.7 Stage VII: Fresh Grouped Execution

The final run uses one 2,097,152-position prompt and two deterministic responses, each with three input and two scored tokens. Rewards $[0, 1]$ produce advantages $[-1, 1]$. The run captures the prompt, materializes both old-policy score sets, then executes two serial 78-layer policy backwards into rank-8 LoRA adapters. After the second backward, workers make one optimizer call and one gradient clear each; all 32 ranks terminate.

PyTorch allocation during prefix capture is not uniform across ranks. The capture-window `max_memory_allocated` ranges from 112.571 to 145.148 GB per rank. A direct read of dsw-6601 reports NVIDIA H20-3e devices with 143,771 MiB (140.401 GiB, or 150.755 GB in decimal) each. The 32-GPU GLM log does not include its own device query; if those workers expose the same per-device total, the capture readings correspond to 74.7–96.3% of device total. This 32.577 GB spread reveals substantial rank nonuniformity and motivates testing better placement and load balance. A historical Qwen log also reports a PyTorch allocator capacity of 139.73 GiB (about 150.0 GB decimal); that process-level limit is not the H20 hardware specification. Because the GLM counter is read before response replay, it does not quantify whole-transaction headroom or establish that any specific context beyond 2M fits. The completed 2M run is an operating point, not an OOM-derived memory frontier; it establishes grouped execution within the fixed budget.

This stage adds no semantic guarantees beyond earlier stages: the DSA response operator is still CP-local, Megatron gradient finalization is skipped for CP-replicated non-expert adapters, the prompt state remains detached, and the supplied responses do not exercise online rollout or reward computation. Those properties are reported beside the terminal evidence in the execution audit that follows on every participating rank.

8 Execution Receipts and Trace Evidence

The acceptance evidence is organized around terminal events and semantic scope. A row enters the main table only when every participating worker reaches the requested scoring, backward, and optimizer-call boundary. The table does not promote those events to a coherent distributed update.

Table 5 Audited fixed-budget execution receipts. Qwen combines a 2,088,960-position prompt with 8,192 response inputs for an exact 2,097,152 context positions; GLM’s prompt alone contains 2,097,152 positions. The fixed allocations are eight H20s for Qwen and 32 H20s for GLM. Terminal evidence records worker-local events, not a coherent distributed parameter update. Hardware and suffix workloads differ, so wall times are not comparable.

Path	Prompt / response	Observed G	Hardware/layout	Terminal evidence on every worker	Wall (s)	Peak GB
Qwen global forward	2,088,960 / 8,192 input*	2	8 H20, CP8	old + ref + backward + local AdamW call	5198.780	97.503
Qwen global forward	2,088,960 / 8,192 input*	8	8 H20, CP8	eight serial members + local AdamW call	6785.225	97.711
GLM CP-local DSA	2,097,152 / 3 input (2 scored)	2	32 H20, TP1/CP32/EP32	two old scores + 2×78-layer backward + optimizer call	2975.138 [†]	n/r

*The Qwen scorer drops the first response label, so 8,192 response input tokens yield at most 8,191 scored positions; the run log does not record the realized count. Peaks are `max_memory_allocated/109` over the measured run. [†]GLM prompt capture plus supplied-group execution; model creation adds 170.211 s. No valid GLM whole-run peak was recorded. The recorded capture-window peak allocation spans 112.571–145.148 GB per rank but is read before response replay, so it remains a diagnostic rather than the GLM Peak entry. All rows condition on a detached prompt state; distributed-update consistency is unverified. The observed G values are validation anchors, not loop limits: both paths are driven by configured member lists rather than hard-coded to those values, while payload and score storage still grows with group size. Only the listed settings have terminal receipts.

8.1 Runtime Ownership and Pinned Rerun Stack

In the current rerun protocol, MinT Runtime supplies the model/session control plane and managed Megatron trainer groups (Mind Lab, 2026). The pinned local stack implements its asynchronous request lifecycle with Ray-resident Megatron workers. LongStraw is an opt-in long-context execution extension on that substrate. It adds GLM prefix capture, CPU-resident MLA/DSA state, CP/EP ownership, response-only replay, serial GRPO accumulation, and source-bound validation and receipt tooling. The 2M runner creates the model through MinT and then invokes a LongStraw-installed method on the resident Megatron actor rather than the stock MinT forward/backward path. Thus MinT owns the managed transaction and worker lifecycle; LongStraw owns the rerun’s long-context execution path.

The accepted historical 2M receipt predates this pinned stack. The current tree implements global cross-CP DSA composition and restores Megatron gradient finalization. Only a 32K forward canary covers those paths; no fresh source-bound 2M rerun or full gradient-parity receipt upgrades the historical evidence.

Table 6 pins the source identities declared by the current rerun protocol. These declarations are prospective: they do not retroactively establish that the historical 2M receipt was produced from this exact stack.

Table 6 Exact source identities declared for the current GLM rerun protocol. They bind future reruns to an auditable stack but do not retroactively bind the historical 2M execution receipt.

Component	Exact identity	Role	Declared source state
MinT Runtime	12c83d904df5faf3e2cd60633b448a8317d84ee0	Model/session control plane and Ray-resident Megatron worker substrate	Clean exact checkout; LongStraw remains an external opt-in extension
Megatron-LM	03db8324007ed7b33edffc147160bebf9552846c	Distributed model, parallelism, gradient, and optimizer runtime	Clean exact checkout
Megatron-Bridge	22edeb2a487d6a9cc0dcea567827826cc76427c2	GLM-5.2 model and LoRA configuration bridge	Exact base plus the bundled GLM-5.2 integration patch
verl	d2916f5a0ed346464d8999e040e0ebb05bb8fadb	Training-datum conversion and MCore integration used by MinT	Exact base plus the bundled MCore compatibility patch
GLM-5.2	b4734de4facf877f85769a911abafc5283eab3d9	Base weights, tokenizer, and model configuration	Exact model snapshot revision

8.2 Qwen Receipts

The Qwen $G = 2$ and $G = 8$ probes share one 2,088,960-position prompt and use 8,192 response-input tokens per member, producing a context length of exactly 2,097,152. Both finish on eight H20 workers. The reported whole-run allocated-memory peaks are 97.503 and 97.711 decimal GB per rank, and wall times are 5198.780 and 6785.225 seconds. The near-flat peak from $G = 2$ to $G = 8$, together with the recorded member ordering, validates the serial response-graph lifetime at these endpoints. Post-prefix work is 271.278 versus 266.476 seconds per member, while mean total wall time per response falls from 2,599.390 to 848.153 seconds because the same prefix is amortized over four times as many members. These are two validation anchors, not a group-size limit or a broadly applicable throughput scaling law.

Every worker records old/reference scoring, policy backward, and a local AdamW call. Forward response attention composes all CP8 KV partitions. The backward audit narrows the terminal claim: dQ is all-reduced, while page-owner dK/dV contributions to replicated adapters are not. No post-step parameter hash, delta, or replica comparison is stored; these are execution, not distributed-update, receipts.

The workload is synthetic. Old and reference scores come from the same current runner, $\beta = 0$, and the implemented live policy term is unclipped. At the first step, old and current scores coincide and the importance ratio is one. The probes do not exercise an independent reference model, nonzero KL, or the clipped-min branch of the GRPO objective in Equation 2.

8.3 GLM Terminal Manifest

The GLM manifest records 32/32 rank reports, two old-policy score phases, two live policy backwards, one optimizer call and one zero-grad call per worker, and no execution errors. The supplied rewards are $[0, 1]$, giving normalized advantages $[-1, 1]$. The per-member losses are in $[-0.5, 0.5]$. The old snapshot is also the

reference for the first update, so measured KL is zero and the importance ratio is one. The code contains a clipped surrogate with $\epsilon = 0.2$, but that branch remains inactive in the live beyond-2M-context transaction.

Model creation takes 170.211 seconds. Prompt capture plus both old phases, both policy phases, and terminal optimizer calls take 2975.138 seconds. These are single-run wall times. There is no variance estimate.

8.4 Trace Inventory

The final GLM run emits separate rank/phase JSONL traces. Table 7 summarizes the inventory.

Table 7 GLM final-run trace inventory. Forward/backward (F/B) counts are per rank within each phase. Across the two policy phases, the audit finds 4,992 forward and 4,992 backward layer-end events, with all 78 layers checkpointed on every policy rank.

Phase	Files	JSONL lines	Layer-end events per rank	Checkpoint coverage
Group 0, old	32	5,120	78 F / 0 B	No policy backward
Group 1, old	32	5,120	78 F / 0 B	No policy backward
Group 0, policy	32	12,672	78 F / 78 B	78 of 78 layers
Group 1, policy	32	12,672	78 F / 78 B	78 of 78 layers
Total	128	35,584	Policy: 4,992 F / 4,992 B	78 layers per policy rank

The 64 old-policy traces cover all 78 forward layers on every rank. The 64 policy traces cover all 78 forward and 78 backward layer events on every rank. All policy layers report activation checkpointing. The traces include attention-projection and sparse-attention backward events. They show that the architecture-shaped layer graph executed twice and that every worker reached the terminal path.

Trace presence is not numerical correctness. A layer-end event does not record the complete parameter-gradient tensor, and the final run stores no numeric gradient norm. Trace completion also cannot reveal a skipped cross-rank reduction. The historical accepted GLM runner bypasses Megatron gradient finalization for CP-replicated non-expert adapters. Thus the trace remains an execution-level receipt.

8.5 Memory Accounting

After model, adapter, optimizer, and input construction, Qwen resets the CUDA peak counter for the measured probe. It reports allocated bytes in decimal GB, excluding reserved blocks and host memory.

The GLM artifact records a capture-window `max_memory_allocated` range of 112.571–145.148 GB per rank. The counter is reset immediately before no-grad prefix capture and read before full-GRPO response replay. The prompt-state contract stores 5.8125 GiB on CPU per rank and stages only 72 MiB for an IndexShare layer or 88 MiB for an index-computing layer, so the length-dependent state is not persistently GPU-resident. The rank spread shows nonuniformity and suggests room for placement and load-balance improvements. Thus 2M is an achieved operating point rather than a measured capacity ceiling. Because the artifact records neither reserved memory nor non-PyTorch usage and omits later replay and checkpoint-recomputation peaks, we retain “n/r” in Table 5 and infer from this capture-only diagnostic neither a whole-transaction limit nor a specific $> 2M$ capacity ceiling for the complete GLM transaction.

8.6 Semantic Claim Matrix

Table 8 maps the receipts to four evidence levels and separates completed from missing checks.

The ordering of the remaining work follows the matrix. Qwen first needs selective model-parallel composition of shard-local K/V and hidden-gradient contributions. The current GLM tree implements global DSA candidate/output composition and restores Megatron gradient finalization, but it still needs short-context full adapter-gradient and optimizer-delta parity followed by a fresh source-bound 2M rerun. Only after these checks does it make sense to build a repeated online rollout, reward, update, checkpoint, and reload loop.

Table 8 Audited evidence matrix. Execution capacity records that the requested program reached its terminal optimizer calls; it does not imply a faithful distributed forward, a synchronized update, or conventional full-gradient parity.

Evidence level	Criterion	Qwen3.6-27B	GLM-5.2
Execution capacity	Requested stages, backwards, and terminal optimizer calls complete	Yes. The fixed-budget 2.097M $G = 2$ and $G = 8$ response-only paths complete on eight H20 GPUs	Yes. All 32 ranks report two 78-layer backwards and terminal optimizer calls
Global response forward	Response tokens use the intended prompt-wide operator	Yes, with a numerical caveat. CP8 performs a global LSE/output merge, but its numerator is BF16	No. DSA top-2048 selection and attention are local to each 65,536-token CP shard; no candidate or output merge executes
Distributed update	Adapter gradients and parameter updates are consistent across the distributed ownership layout	No. Required reductions for shard-local K/V adapter gradients are missing, and AdamW is called per rank	No. The replay path bypasses <code>finalize_model_grads</code> for CP-replicated non-expert adapters
Full-gradient parity	Every trainable gradient matches a conventional full-sequence numerical reference	No. The prompt state is detached and no numerical full-sequence comparison exists	No. The prompt is detached, no numerical reference exists, and response DSA already differs in the forward pass

9 Fixed-Budget Systems Lessons

The two implementations share a prompt-state graph boundary but expose different bottleneck chains, determined by what each architecture must retain and communicate afterward.

9.1 Fixed-Budget Capacity Comes from Lifetime, Not Sparsity Alone

Neither path avoids the long prompt forward, and neither path adds accelerators as context grows within its reported envelope. Every prompt token still passes through every decoder layer. The capacity gain comes from allowing prompt attention scratch, dense FFN intermediates, MoE routes, expert-token permutations, and adapter activations to die before response backward. Once the prompt has been captured, the live autograd working set follows response length rather than prompt length.

This explains why several plausible optimizations are insufficient on their own. DSA reduces attention arithmetic but retains long-context indexing (DeepSeek-AI, 2025; Bai et al., 2026). MoE evaluates only selected experts but expands tokens into routed rows and distributes a very large parameter set (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). QLoRA reduces persistent trainable state, not activations (Dettmers et al., 2023). Activation checkpointing reduces saved tensors by trading storage for recomputation (Chen et al., 2016; Korthikanti et al., 2022). In the traced runs reported here, recomputation still re-created peak workspace at the guarded boundary. The working design composes all of these tools around the prompt-state graph boundary shared by both execution paths under review.

9.2 Physical Ownership Is Part of the Algorithm

A context shard is useful only when its physical allocation is also sharded. The first Qwen page implementation retained small slices whose parent chunks remained allocated. The logical page table looked distributed while allocator memory did not change. Copying selected pages into right-sized tensors made the ownership statement true and moved the 2,088,960-position prefix peak into the feasible range.

The same principle applies to GLM CPU state. A page tensor, its position order, and the consuming layer form one object. Restoring correct bytes in the wrong CP order changes causal positions and sparse selection. Offload transfers operator state and must preserve identity and order.

9.3 Dense and MoE Move the Peak to Different Places

For Qwen, every token evaluates the same dense FFN, and no expert routing state crosses devices. Context-growing storage is concentrated in 16 full-attention KV sets; 48 GDN boundaries remain fixed-size with prompt length. Page compaction and context partitioning therefore target the dominant stored state.

Observed numeric envelope.

- **Fixed accelerator budgets:** Qwen uses eight H20 GPUs in its reported 2M and 4.25M paths; the GLM grouped 2M path uses 32 H20 GPUs. Device count is an input constraint, not the reported scaling axis.
- **Context receipts:** Qwen executes $2,088,960 + 8,192 = 2,097,152 = 2^{21}$ positions per response, exactly $8\times$ its stored native setting. GLM captures a 2,097,152-position prompt, $2\times$ its published setting.
- **Configured group:** Qwen $G = 2 \rightarrow 8$ changes peak allocation by +0.208 GB (+0.213%) while adding 1,586.445 s; one prompt capture is amortized across all serial members.
- **Measured-phase accelerator-hours:** elapsed time times allocated devices gives 11.553 and 15.078 H20-hours for Qwen 2M ($G=2$) and ($G=8$), 48.334 for the 4.25M resident replay, and 26.446 for the GLM grouped path. These are allocation receipts, not utilization, energy, or total-cost estimates.
- **Qwen capacity bracket:** Qwen’s physical KV arithmetic adds 8.192 decimal GB per rank per one million additional global prompt positions. A train-block proxy passes at 4,538,368 and OOMs one 4,096-position chunk later, but a fuller path already OOMs in policy backward at 4,456,448.
- **GLM 2M is not a measured frontier:** GLM retains 5.8125 GiB of CPU prompt state per rank and stages 72–88 MiB per layer on GPU. The recorded capture-window peak allocation ranges from 112.571 to 145.148 GB per rank, revealing a 32.577 GB rank spread and motivating better-balancing tests. Because this counter precedes response replay, it is a rebalancing diagnostic rather than a whole-transaction headroom or $> 2M$ capacity claim.

For GLM, expert sparsity separates total parameters from activated parameters, but full-sequence training still handles eight expert assignments per token. At the final CP shard size, one expanded BF16 hidden buffer can be 6 GiB before output, permutation, or LoRA work. Removing DSA scratch simply revealed the next MoE allocation. Response-only replay succeeds because only a few suffix rows are routed under autograd; the no-grad prompt routes remain transient.

The broader lesson is that sparsity transfers cost. DSA turns dense attention into an index-selection and selected-value movement problem. MoE turns dense FFN compute into expert residency and token communication. A training system must implement the transferred problem, not only count fewer FLOPs.

9.4 Context and Expert Parallelism Are Orthogonal

Context parallelism partitions token history. Expert parallelism partitions FFN parameters (Liu et al., 2025). Folding CP32 and EP32 onto the same 32 ranks is a useful placement, but their collectives have different meanings. EP all-to-all sends response rows to expert owners and combines expert outputs. It cannot turn 32 local sparse candidate sets into one global top-2048 set. A CP attention merge cannot balance routed expert load. This distinction also appears in backward. A globally composed attention forward does not guarantee a coherent adapter update. Replicated projections need all shard-local gradient contributions. In the inspected Qwen path, dQ is all-reduced, while dK/dV and the corresponding adapter gradients remain local to page owners; eight independent AdamW instances then step. A complete distributed implementation must specify parameter ownership and gradient reduction as carefully as forward state ownership.

9.5 Forward Fidelity, Update Consistency, and Gradient Parity Differ

The Qwen and GLM paths occupy different positions on the four-level evidence ladder.

- Both produce fixed-budget 2.097M execution receipts: the requested scoring, response backward, and terminal optimizer calls complete on every worker.
- Qwen composes a global full-attention response forward over CP8. Its production numerator reduction uses BF16, so the claim is partition-correct forward semantics, not bitwise FP32 equality. GLM response DSA remains local to one CP shard.
- Qwen distributed-update consistency is incomplete because shard-local K/V adapter gradients are not synchronized. The GLM receipt also requires an explicit reduction: its custom resident path bypasses

`finalize_model_grads`, so CP-replicated attention, dense/shared FFN, and output-head adapter gradients remain local before the optimizer call.

- Neither path matches a conventional full-sequence gradient because the prompt state is detached. GLM must first close its forward mismatch; both then require a parameter-by-parameter gradient reference at a shorter context.

Loss values and layer-end backward events cannot substitute for these tests. A single global gradient norm could also hide target-family errors. The parity suite must compare each LoRA shard, report per-family outliers, compute global cosine and relative L_2 , and verify the optimizer delta from identical initial state.

9.6 Group Scaling Is a Scheduling Result

Qwen $G = 8$ consumes nearly the same peak allocated memory as $G = 2$, consistent with response branches being serialized after one prompt capture. A fourfold group increase raises measured peak allocation by 0.208 GB, or 0.213%, while post-prefix work grows by a factor of 3.93. The marginal time across the six additional members is 264.408 seconds per member. Meanwhile, sharing the 4,655-second prefix reduces mean total wall time per supplied response by 67.4%, from 2,599.390 to 848.153 seconds.

The implementation is therefore parameterized by group cardinality rather than capped at the observed $G = 2$ or $G = 8$. The serial loop consumes a configured member list and is not specialized to either value, but only these two settings have Qwen receipts. This does not make memory strictly constant: stored inputs, labels, rewards, reports, and frozen scores grow with G or total response length, and larger groups remain subject to the wall-time budget. Nor does it make G algorithmically irrelevant; changing the group changes reward normalization and the GRPO estimator. The measured claim is narrower: group size is not the dominant live-autograd capacity axis in this design. The GLM acceptance run uses two extremely short responses and has no whole-run peak, so it provides no independent group-scaling curve.

These single-run timings span different models, GPU counts, suffixes, numerical paths, and semantic gaps. They establish termination within a specified resource envelope, not a model or system throughput ranking.

9.7 Eight H20s Carry Qwen to a 4.25M Context Envelope

Under the same eight-H20 budget used for the 2,097,152-position receipts, Qwen reaches 4.25M context, where 4.25M means exactly $4.25 \times 2^{20} = 4,456,448$ positions: 4,448,256 prompt positions and an 8,192-position response. A resident $G = 8$ run completes one 1,086-chunk prefix capture, all eight serial old/reference/policy branches, and all four 2,048-position response-backward blocks per member. The measured path takes 21,750.133 seconds from prefix start through the group and peaks at 82.960 GB per rank. This is a complete 4.25M response-replay receipt, not a prefix-only or stage-1-forward result.

The stronger prefix-frozen response-only run converts that receipt into an 8-step curve. Each step accumulates all eight group members before applying an optimizer update. Across 64 member replays, every rank records eight applied optimizer steps, `prefix_stale=false`, and a peak of 83.894 GB. Freezing adapter deltas on prompt positions makes the captured prefix invariant across those updates, so this is complete continuous training for the stated prefix-frozen objective. It is not the original prompt-adapted QLoRA objective, whose state must be recaptured after every optimizer step before reuse.

The remaining frontier numbers are capacity diagnostics rather than the main 4.25M result. A clean prompt-adapted run before detached-prefix gradient-page pruning OOMed in policy backward. A train-block proxy reaches 4,538,368 before the next 4,096-position chunk OOMs at 4,542,464. None of these labels erases the separate CP8 reduction gap described above: local optimizer application is recorded, but coherent replicated-adapter updates still require the missing cross-rank gradient composition. See Appendix F for full details.

9.8 The Claim Is a Fixed-Budget Envelope, Not a Context Record

Million-token execution predates LongStraw, and this report makes no first, longest, or world-record claim. Its comparison axis is the fixed accelerator envelope: 4,456,448 Qwen positions on eight H20 GPUs and 2,097,152 GLM positions on 32 H20 GPUs. Adding devices is a valid scale-out strategy, but it is outside

these experiments. LongStraw instead trades GPU residency for compact physical pages, CPU state, recomputation, and serial replay time within the stated allocations without adding ranks.

The stored Qwen configuration has a native maximum position setting of 262,144; its receipt at a context length of 2,097,152 is exactly $8\times$ that setting. The published GLM-5.2 configuration uses 1,048,576; the captured 2,097,152-position prompt is $2\times$ that setting. No long-context task evaluation, loss curve, or repeated learning result accompanies the receipts. The result is therefore an accelerator-bounded systems feasibility envelope, not evidence of useful learned behavior or a context-length leaderboard.

10 Related Work

10.1 Scale-Out Long Context and the Fixed-Budget Axis

Prior work already establishes that million-token sequence processing is possible. Ring Attention reports exact-attention training at 4.096M positions for a 7B model on 32 A100 GPUs (Liu et al., 2023). DeepSpeed-Ulysses studies a scale-out regime in which sequence length and device count grow together; its experiments scale to 256 A100 GPUs and include a one-million-token sequence for a 1.2B GPT model (Jacobs et al., 2023). ByteScale reports a 2M LLaMA-7B case on 1,024 GPUs within a production cluster exceeding 12,000 GPUs (Ge et al., 2025). USP then combines ring-style and all-to-all sequence parallelism and analyzes its interaction with tensor parallelism, ZeRO, recomputation, and offload (Fang and Zhao, 2024). DistFlashAttn adds load-balanced exact-attention scheduling and overlaps peer-to-peer KV transfer with attention compute, while LoongTrain combines head and context parallelism through 2D-Attention and a double-ring schedule (Li et al., 2023; Gu et al., 2024). Both remain scale-out methods that widen device-level parallelism.

Ring Attention and ByteScale provide stronger full-sequence training semantics; LongStraw uses fewer devices for a different detached-prefix GRPO replay problem. The comparison therefore motivates the budget and evidence axes rather than an efficiency ratio across unmatched workloads.

Large-scale technical reports operate at a different industrial scale. DeepSeek-V3 reports a 2,048-H800 training cluster and 2.788 million H800 GPU-hours for its full 671B-MoE training program (DeepSeek-AI, 2024b). LongCat-Flash reports a 560B MoE trained with infrastructure spanning tens of thousands of accelerators, while GLM-5 reports a 744B MoE trained over 28.5 trillion tokens (Meituan LongCat Team, 2025; GLM-5-Team, 2026). These results are not apples-to-apples baselines for LongStraw: model size, objective, hardware, sequence semantics, and evidence level all differ. They instead establish why our claim is not “first long context.” LongStraw fixes the accelerator envelope at eight H20 GPUs for Qwen and 32 H20 GPUs for GLM, then asks which state-lifetime and ownership decisions make a GRPO-shaped execution path fit. We report a budget-conditioned feasibility envelope, not a context-length or throughput leaderboard. The relevant comparison is accelerator-bounded GRPO execution, not absolute long-context capability.

10.2 Relationship to MinT

MinT manages LoRA adapter revisions across rollout, update, export, evaluation, and serving over resident base-model deployments; its training plane includes distributed Megatron execution for dense and MoE models, parallelism-aware adapter state, and MLA/DSA support (Mind Lab, 2026). The current LongStraw rerun interface directly reuses that managed control plane and resident Megatron LoRA substrate, but changes the state boundary inside one update: it captures architecture-specific prompt state without autograd, stages that state under CP/EP ownership, and serially rebuilds response graphs. This architectural lineage does not retroactively bind historical LongStraw receipts to the current MinT revision. The one-million-adapter result of MinT measures addressable policy-catalog scale with bounded serving working sets, not long-context execution. The 2M and 4.25M quantities in LongStraw count positions in execution receipts, not addressable adapter revisions or simultaneously served policies.

10.3 Memory-Efficient and Distributed Attention

Exact attention does not require materializing the full quadratic score matrix. Memory-efficient algorithms stream score blocks or improve IO locality while preserving the softmax result (Rabe and Staats, 2021; Dao

et al., 2022). Ring Attention distributes sequence blocks over devices and composes attention as the blocks circulate (Liu et al., 2023). DeepSpeed-Ulysses exchanges sequence and attention-head partitions with all-to-all collectives, while USP combines Ulysses-style and ring-style sequence parallelism (Jacobs et al., 2023; Fang and Zhao, 2024). These methods address the core attention computation. Our setting adds a training-specific boundary: prompt state is retained across old, reference, and policy branches, while the suffix is replayed under autograd. The central questions become physical page ownership, state validity, and which distributed reductions are required in both forward and backward.

PagedAttention makes KV allocation and page tables first-class serving-system objects (Kwon et al., 2023). In this report, pages cross a training boundary. A logical page shard must own a compact physical allocation; a view into a larger parent chunk does not release memory. Prompt pages are read-only during grouped response replay, and their validity ends when an optimizer step changes the adapted parameters.

10.4 MLA, Sparse Attention, and Index Reuse

MLA compresses KV state into a latent (DeepSeek-AI, 2024a). DeepSeek-V3.2 forms DeepSeek Sparse Attention (DSA) by adding a lightweight top-k indexer to MLA (DeepSeek-AI, 2025). The GLM-5 report provides MLA/DSA background (GLM-5-Team, 2026); the pinned GLM-5.2 configuration artifact instantiates index producers and IndexShare consumers (zai-org, 2026). IndexCache formalizes the generic cross-layer sparse-index reuse mechanism (Bai et al., 2026); the pinned configuration supplies this model’s concrete pattern.

Our focus is the resulting training-state contract. A saved MLA latent page is not sufficient when the sparse indexer also needs long-context keys. Index reuse saves work but creates producer/consumer lifetime inside each response forward. Local sparse selection is not global selection over a context-parallel prompt. Distributed training must define candidate merge, selected-value movement, and output composition.

10.5 MoE Training and Multidimensional Parallelism

Sparse MoE models distribute the full parameter set but activate only selected experts per token (Lepikhin et al., 2021; Fedus et al., 2022). Expert parallelism reduces per-rank parameter residency but introduces token permutation, all-to-all dispatch, expert computation, and combine. DeepSeek-V3 links fine-grained MoE to cross-node overlap and memory-efficient training (DeepSeek-AI, 2024b). LongCat-Flash similarly co-designs MoE layer structure, communication overlap, deterministic kernels, and its EP/CP/PP layout (Meituan LongCat Team, 2025). MoE Parallel Folding analyzes heterogeneous tensor, context, expert, data, and pipeline mappings (Liu et al., 2025). Tutel selects MoE all-to-all implementations and pipeline degree by workload and cluster scale; MegaBlocks maps irregular expert-token work to block-sparse operations (Hwang et al., 2022; Gale et al., 2022). These systems optimize sparse kernels, communication, and parallel mappings; LongStraw instead isolates long-lived prompt state from each short-lived native MoE replay and makes the remaining distributed-gradient obligations explicit before optimizer consumption.

The GLM path in this report uses CP32 and EP32 over the same ranks. This placement is economical, but the parallel dimensions remain semantically different. CP owns context and must preserve attention; EP owns experts and routed-token computation. Neither collective can replace the other.

10.6 Distributed Training State and Optimizer Sharding

Megatron-LM established tensor model parallelism for large Transformers (Shoeybi et al., 2019); subsequent Megatron training systems compose tensor, pipeline, and data parallelism at cluster scale (Narayanan et al., 2021). ZeRO and PyTorch FSDP instead shard parameters, gradients, and optimizer state across data-parallel workers (Rajbhandari et al., 2020; Zhao et al., 2023). These systems reduce persistent state and define ownership for standard layer graphs. LongStraw composes CP, EP, and sharded optimizer machinery with custom prompt-state replay. That boundary must still call the appropriate gradient finalization or selective reduction; optimizer sharding cannot recover contributions that never reach a parameter owner.

Heterogeneous-memory variants widen the placement space. ZeRO-Offload moves selected model states from GPU to CPU memory, while ZeRO-Infinity can place partitioned model states in CPU or NVMe memory

(Ren et al., 2021; Rajbhandari et al., 2021). LongStraw’s CPU prompt pages are different state with a different lifetime, but they inherit the same requirement that ownership and transfer timing be explicit.

10.7 Activation Checkpointing and Parameter-Efficient Adaptation

Activation checkpointing trades recomputation for lower retained activation memory (Chen et al., 2016). For the GLM response path, the useful boundary is the complete decoder layer. Checkpointing attention alone leaves MoE router, dispatch, selected-expert, and adapter intermediates alive. The long prompt is not checkpointed for backward; it is evaluated without autograd and represented by stored conditional state. Selective activation recomputation instead retains the layer boundary while recomputing only memory-heavy, relatively inexpensive attention operations (Korthikanti et al., 2022). That complementary design reduces redundant recompute; it does not replace the full-layer boundary required by our native MoE replay.

LoRA and QLoRA reduce trainable parameter, gradient, and optimizer-state storage (Hu et al., 2021; Dettmers et al., 2023). They do not remove response activations or the need to synchronize replicated adapter gradients. In Qwen, full-attention forward composition is global, but coherent CP8 adapter updates still require the missing cross-rank gradient synchronization in Section 5. AdaLoRA allocates adapter rank under a parameter budget, while DoRA separates weight magnitude from low-rank directional updates (Zhang et al., 2023; Liu et al., 2024). Related work explores hierarchical rank allocation, intra/inter-layer adapter sharing, mixed-precision fidelity, and joint quantization with low-rank adapters (Zhou et al., 2025a,e, 2026a,b). GPTQ and SparseGPT are established post-training quantization and pruning methods (Frantar et al., 2022; Frantar and Alistarh, 2023); global rank/sparsity optimization and probabilistic layer quantization further change the base model’s storage and sensitivity profile (Zhou et al., 2025b,c). Qwen uses NF4 QLoRA and GLM rank-8 LoRA; adapter design is not a LongStraw contribution. LongLoRA combines LoRA with shifted sparse attention for efficient long-context adaptation (Chen et al., 2023b); LongStraw instead preserves the model’s native prompt semantics and changes graph lifetime and placement.

10.8 Adapter Serving and Inference Infrastructure

Punica and S-LoRA establish multi-tenant batching, memory management, and kernel paths for serving many adapters concurrently (Chen et al., 2023a; Sheng et al., 2023). Dynamic operator selection likewise treats adapter placement and operator reuse as serving-time systems problems (Zhou et al., 2025d). Budget-driven depth routing and dynamic low-rank substitution address adaptive inference, where the objective is to reduce latency or compute for a fixed request (Zhou et al., 2026d,c). LongStraw is adjacent but distinct: it targets the training-time prompt/response boundary, preserves architecture-specific state across a grouped GRPO update, and exposes the gradient-ownership conditions that serving systems do not need to satisfy.

10.9 GRPO Systems

PPO alternates policy sampling with multiple optimization epochs over a clipped surrogate objective (Schulman et al., 2017). GRPO normalizes outcome rewards within a response group and removes the learned critic used by PPO-style training (Shao et al., 2024). DeepSeek-R1-Zero applies GRPO in a reasoning-training pipeline that begins without supervised fine-tuning (DeepSeek-AI et al., 2025). DAPO then extends the GRPO family with decoupled clipping, dynamic sampling, token-level policy-gradient loss, and overlong reward shaping (Yu et al., 2025). These methods change optimization behavior, not prompt-state lifetime.

DeepSeek-V3.2 describes additional stabilization mechanisms for scaled GRPO, including off-policy sequence masking and preservation of MoE routing between inference and training (DeepSeek-AI, 2025). LongCat-Flash-Thinking-2601 develops an asynchronous system for long-tailed environment interaction and large-scale agentic RL (Meituan LongCat Team, 2026). DeepSpeed-Chat, HybridFlow, and OpenRLHF address the broader RLHF execution problem, including model-role placement and transitions among generation, scoring, and training; OpenRLHF assigns rollout and actor/training engines distinct roles under Ray (Yao et al., 2023b; Sheng et al., 2024; Hu et al., 2024). AReaL goes further by decoupling rollout and training asynchronously and explicitly managing data staleness (Fu et al., 2025).

Our acceptance workload is narrower. Responses and rewards are supplied, and the runs terminate after one update-shaped transaction. The report isolates the long-context policy-backward machinery from rollout

scheduling, environment execution, and repeated learning. This separation permits precise systems receipts, but it also prevents claims about online RL throughput or downstream policy improvement.

11 Conclusion

LongStraw shows that long-context GRPO under a fixed GPU budget is a tensor-lifetime and ownership problem, not a context-length race or a single-kernel change. Long-context processing itself is established when a sufficiently large accelerator fabric is available. The systems question here is how much GRPO-shaped execution fits without adding accelerators. LongStraw fixes the inventory at eight H20 GPUs for Qwen and 32 H20 GPUs for GLM, then makes the contract among model structure, prompt-state ownership, suffix replay, parallel communication, gradient composition, and optimizer ordering explicit. The shared mechanism is a no-grad prompt boundary followed by serial short-response replay; the state on that boundary is architecture specific at every layer and ownership boundary in both model families.

For the dense-hybrid Qwen model, compact physical KV pages and recurrent GDN state make the prompt fit across CP8, and a global LSE/output reduction composes the full-attention response forward. For GLM, CPU-resident MLA and indexer-key pages, one-layer staging, complete-layer checkpointing, IndexShare reconstruction, and CP32/EP32 placement carry two 78-layer response backwards to terminal distributed-optimizer calls. The GLM progression from 32K full-graph failures through prefix capture, layer-0 replay, all-layer closure, and grouped execution shows why capacity must be built one dependency class at a time.

The same audit also identifies what did not run correctly. Qwen lacks composition of shard-local K/V adapter gradients. The historical GLM receipt uses CP-local DSA and bypasses Megatron gradient finalization for CP-replicated non-expert adapters. Both detach the prompt state. The durable result is therefore a budget-conditioned, architecture-aware execution envelope rather than a first-to-length or longest-context claim. Within the same eight-H20 envelope, Qwen completes a 4.25M $G = 8$ response replay and, under the explicit prefix-frozen response-only parameterization, eight consecutive $G = 8$ optimizer steps comprising 64 member replays at a peak of 83.894 GB per rank. The 4,538,368/4,542,464 train-block bracket exposes further capacity room; the 32-H20 GLM path carries a 2,097,152-position prompt through two complete 78-layer backwards and terminal optimizer calls. That GLM point is not an OOM-derived frontier: recorded capture-window peak allocation ranges from 112.571 to 145.148 GB per rank, and no GLM probe above 2M was attempted. The rank spread suggests room for better placement and load balance, but only a direct $> 2M$ probe with whole-transaction memory accounting can turn that opportunity into a capacity result. These are accelerator-bounded feasibility receipts, not minimum-cost or distributed-gradient-parity claims.

12 Limitations and Validation Roadmap

These rank-complete execution receipts lack support for stronger correctness or training claims.

12.1 Distributed Gradient Composition Is Incomplete

The Qwen forward path composes global CP8 attention statistics, and its custom backward all-reduces dQ . It leaves page-owner dK/dV contributions local even though the K/V projection adapters are replicated. No DDP or selective model-parallel gradient reducer runs before eight independent AdamW calls. Consequently the adapter updates can diverge and are not the gradient of the global conditional response computation.

The historical GLM custom resident path calls `loss.backward()` outside the normal Megatron schedule and then invokes the distributed optimizer. It bypasses `finalize_model_grads` and the DDP `finish_grad_sync` it normally calls, leaving CP-replicated attention, dense/shared FFN, and output-head adapter gradients unreduced. Routed-expert adapters differ: EP32 with expert data-parallel size one makes them unique shards that need no averaging; an optimizer all-gather can still align replicas after updates from incorrect local CP gradients. The current tree restores Megatron gradient finalization, but it has no fresh source-bound 2M receipt or full adapter-gradient and optimizer-delta parity.

Before a stronger fixed-budget training claim, both paths must record the expected gradient collective, compare post-step parameter hashes across ranks, and verify the optimizer delta against a reference.

12.2 The Historical GLM Receipt Uses CP-Local Response Attention

In the historical receipt, every GLM rank selects top-2048 positions from its own 65,536-token prompt shard. There is no cross-rank candidate merge, global top-2048 selection, selected-value exchange, or output composition. IndexShare reuses this local schedule. It is an architecture-shaped surrogate, not distributed DSA over a global prompt of 2,097,152 context positions as seen by each response query.

The current tree implements local candidate production, global top-2048 selection, selected-value movement, and sparse-output composition. It still needs per-layer response-logit parity against an unsharded reference at a context that fits on one process, followed by a fresh source-bound 2M rerun.

12.3 The Prompt-State Gradient Is Detached

Even after the distributed forward and gradient collectives are repaired, both paths compute the response gradient conditional on a stored prompt state. They omit the second term in Equation 3. Full-sequence parity must be tested at 32K or 64K, where a conventional reference fits. The comparison must cover every LoRA target family, not only loss, one gradient norm, or a subset of attention projections.

12.4 The Workload Is an Execution Probe

The runs use supplied synthetic responses and deterministic rewards. They do not include policy sampling, environment or reward-model execution, data filtering, checkpoint publication, reload, or repeated updates. The Qwen probe uses the same runner for old and reference scores, $\beta = 0$, and an unclipped live ratio term. The GLM path implements clipping and $\beta = 0.01$, but the first-step ratio is one and old equals reference, so clipping is inactive and KL is zero. Neither run measures policy improvement.

The stored model configurations also have native context settings below the exercised scale: 262,144 for Qwen and 1,048,576 for GLM. No task evaluation establishes useful behavior at this systems scale.

12.5 Resource and Reproducibility Gaps

The fixed-budget claim covers accelerator count and reported device-memory measurements, not total cost. We lack a matched baseline on the same models, objective, suffixes, precision, and hardware, and the artifacts do not fully account for host memory, network traffic, utilization, energy, or monetary cost. LongStraw is therefore accelerator-bounded or resource-constrained, not a claim of universal lowest-cost execution.

All timings are single runs. Qwen samples only $G = 2$ and $G = 8$; GLM has one grouped $G = 2$ receipt, while its $G = 1$ point is only a control-flow canary. There is no broad group-size sweep, evidence of strictly linear time, or proof that total memory is independent of G . Qwen records whole-run allocated CUDA peaks, while the recorded GLM capture-window peak allocation is capture-only and cannot be used as a whole-transaction number. The final GLM artifact does not bind a numeric learning rate, final model revision, token checksum, LoRA seed, software/NCCL version set, whole-step finite scan, final gradient norm, router histogram, or all-to-all byte/timing profile. These omissions do not invalidate terminal execution, but they limit independent reproduction and end-to-end performance analysis across runs.

12.6 Validation Order

The next work should proceed in dependency order.

1. Restore the missing Qwen K/V reductions and validate the current GLM gradient-finalization path at 32K; verify post-step adapter equality across ranks.
2. Validate the current global cross-CP DSA candidate selection, selected-value movement, and output composition with layerwise forward parity.
3. At 32K–64K, compare every LoRA gradient shard and optimizer delta with a conventional full-sequence execution under identical tokens and rewards.

4. Run a real rollout group through sampling, reward computation, frozen old log-probabilities, update, checkpoint, and reload.
5. Repeat the 2M capacity run from the pinned stack, then run a direct > 2M GLM sweep with whole-transaction memory accounting, variance, and phase-separated time.

References

- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL <https://aclanthology.org/2023.emnlp-main.298/>.
- 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.
- Lequn Chen, Zihao Ye, Yongji Wu, et al. Punica: Multi-tenant LoRA serving. *arXiv preprint arXiv:2310.18547*, 2023a. URL <https://arxiv.org/abs/2310.18547>.
- Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.
- 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. *arXiv preprint arXiv:2309.12307*, 2023b. URL <https://arxiv.org/abs/2309.12307>.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.
- DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.
- DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.
- DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023.
- Jiarui Fang and Shangchun Zhao. USP: A unified sequence parallelism approach for long context generative AI. *arXiv preprint arXiv:2405.07719*, 2024. URL <https://arxiv.org/abs/2405.07719>.
- 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.
- Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023. URL <https://arxiv.org/abs/2301.00774>.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. URL <https://arxiv.org/abs/2210.17323>.
- Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025. URL <https://arxiv.org/abs/2505.24298>.
- Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. *arXiv preprint arXiv:2211.15841*, 2022. URL <https://arxiv.org/abs/2211.15841>.
- 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.

- Diandian Gu, Peng Sun, Qinghao Hu, Ting Huang, Xun Chen, Yingdong Xiong, Guoteng Wang, Qiaoling Chen, Shangchun Zhao, Jiarui Fang, Yonggang Wen, Tianwei Zhang, Xin Jin, and Xuanzhe Liu. LoongTrain: Efficient training of long-sequence LLMs with head-context parallelism. *arXiv preprint arXiv:2406.18485*, 2024. URL <https://arxiv.org/abs/2406.18485>.
- 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. *arXiv preprint arXiv:2106.09685*, 2021.
- Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Zilin Zhu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Weikai Fang, Xianyu, Yu Cao, Haotian Xu, and Yiming Liu. OpenRLHF: An easy-to-use, scalable and high-performance RLHF framework. *arXiv preprint arXiv:2405.11143*, 2024. URL <https://arxiv.org/abs/2405.11143>.
- Changho Hwang, Wei Cui, Yifan Xiong, et al. Tutel: Adaptive mixture-of-experts at scale. *arXiv preprint arXiv:2206.03382*, 2022. URL <https://arxiv.org/abs/2206.03382>.
- 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>.
- Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *arXiv preprint arXiv:2205.05198*, 2022. URL <https://arxiv.org/abs/2205.05198>.
- 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, Hyukjoong 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 LLM training. *arXiv preprint arXiv:2310.03294*, 2023. URL <https://arxiv.org/abs/2310.03294>.
- 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:2504.14960v1*, 2025. URL <https://arxiv.org/abs/2504.14960v1>.
- Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring Attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, et al. DoRA: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024. URL <https://arxiv.org/abs/2402.09353>.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://arxiv.org/abs/1711.05101>.
- Meituan LongCat Team. LongCat-Flash technical report. *arXiv preprint arXiv:2509.01322*, 2025.
- Meituan LongCat Team. LongCat-Flash-Thinking-2601 technical report. *arXiv preprint arXiv:2601.16725*, 2026.
- Mind Lab. MinT: Managed infrastructure for training and serving millions of LLMs, 2026. URL <https://arxiv.org/abs/2605.13779>.
- Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using Megatron-LM. *arXiv preprint arXiv:2104.04473*, 2021. URL <https://arxiv.org/abs/2104.04473>.
- Qwen Team. Qwen3.6-27B configuration. Hugging Face model configuration, 2026. URL <https://huggingface.co/Qwen/Qwen3.6-27B/blob/6a9e13bd6fc8f0983b9b99948120bc37f49c13e9/config.json>. Accessed July 16, 2026.
- Markus N. Rabe and Charles Staats. Self-Attention does not need $O(n^2)$ memory. *arXiv preprint arXiv:2112.05682*, 2021.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2020. doi: 10.1109/SC41405.2020.00024. URL <https://arxiv.org/abs/1910.02054>.

- Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. ZeRO-Infinity: Breaking the GPU memory wall for extreme scale deep learning. *arXiv preprint arXiv:2104.07857*, 2021. URL <https://arxiv.org/abs/2104.07857>.
- Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing billion-scale model training. *arXiv preprint arXiv:2101.06840*, 2021. URL <https://arxiv.org/abs/2101.06840>.
- 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.
- Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL <https://arxiv.org/abs/2002.05202>.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. URL <https://arxiv.org/abs/1701.06538>.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, et al. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024. URL <https://arxiv.org/abs/2409.19256>.
- Ying Sheng, Shiyi Cao, Dacheng Li, et al. S-LoRA: Serving thousands of concurrent LoRA adapters. *arXiv preprint arXiv:2311.03285*, 2023. URL <https://arxiv.org/abs/2311.03285>.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. URL <https://arxiv.org/abs/1909.08053>.
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL <https://arxiv.org/abs/2104.09864>.
- 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>.
- 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*, 2023a. URL <https://arxiv.org/abs/2210.03629>.
- Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, et al. DeepSpeed-Chat: Easy, fast and affordable RLHF training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320*, 2023b. URL <https://arxiv.org/abs/2308.01320>.
- Qiyang 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 configuration. Hugging Face model configuration, 2026. URL <https://huggingface.co/zai-org/GLM-5.2/blob/b4734de4facf877f85769a911abafc5283eab3d9/config.json>. Accessed July 16, 2026.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, et al. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023. URL <https://arxiv.org/abs/2303.10512>.
- Yanli Zhao, Andrew Gu, Rohan Varma, et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. URL <https://arxiv.org/abs/2304.11277>.
- Changhai Zhou, Shijie Han, Lining Yang, Yuhua Zhou, Xu Cheng, Yibin Wang, and Hongguang Li. RankAdaptor: Hierarchical rank allocation for efficient fine-tuning pruned LLMs via performance model. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5796–5810. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-naacl.321. URL <https://aclanthology.org/2025.findings-naacl.321/>.
- Changhai Zhou, Qian Qiao, Weizhong Zhang, and Cheng Jin. Large language model compression with global rank and sparsity optimization, 2025b. URL <https://arxiv.org/abs/2505.03801v1>.
- Changhai Zhou, Yuhua Zhou, Yibin Wang, Shijie Han, Qian Qiao, and Hongguang Li. QPruner: Probabilistic decision quantization for structured pruning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4276–4286, Albuquerque, New Mexico, apr 2025c. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.240. URL <https://aclanthology.org/2025.findings-naacl.240/>.

- Changhai Zhou, Yuhua Zhou, Shiyang Zhang, Yibin Wang, and Zekai Liu. Dynamic operator optimization for efficient multi-tenant LoRA model serving. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(21):22910–22918, 2025d. doi: 10.1609/aaai.v39i21.34453. URL <https://ojs.aaai.org/index.php/AAAI/article/view/34453>.
- Changhai Zhou, Shiyang Zhang, Yuhua Zhou, Jun Gao, Qian Qiao, Shichao Weng, Weizhong Zhang, and Cheng Jin. Balancing fidelity and plasticity: Aligning mixed-precision fine-tuning with linguistic hierarchies. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 15885–15896. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.findings-acl.779. URL <https://aclanthology.org/2026.findings-acl.779/>.
- Changhai Zhou, Shiyang Zhang, Yuhua Zhou, Qian Qiao, Jun Gao, Cheng Jin, Kaizhou Qin, and Weizhong Zhang. AutoQRA: Joint optimization of mixed-precision quantization and low-rank adapters for efficient LLM fine-tuning, 2026b. URL <https://arxiv.org/abs/2602.22268>.
- Yuhua Zhou, Ruifeng Li, Changhai Zhou, Fei Yang, and Aimin Pan. BSLoRA: Enhancing the parameter efficiency of LoRA with intra-layer and inter-layer sharing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 78883–78902. PMLR, 2025e. URL <https://proceedings.mlr.press/v267/zhou25k.html>.
- Yuhua Zhou, Shichao Weng, Changhai Zhou, Yuhua Wu, Qian Qiao, Jun Gao, Fei Yang, and Aimin Pan. Deputy: Accelerating large language model inference with dynamic low-rank substitution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 19791–19810, jul 2026c. doi: 10.18653/v1/2026.findings-acl.991. URL <https://aclanthology.org/2026.findings-acl.991/>.
- Yuhua Zhou, Shaoqi Yu, Shichao Weng, Changhai Zhou, Mingze Yin, Fei Yang, and Aimin Pan. BUDDY: BUdget-Driven DYnamic depth routing for adaptive large language model inference, 2026d. URL <https://arxiv.org/abs/2606.09514>.

A Model and Run Configuration

The two tables below bind each architecture to the inspected model snapshot and execution receipt used throughout this report. They separate architecture facts from run parameters and from semantic caveats at the update boundary. A field omitted from the canonical receipt is treated as unbound rather than reconstructed from an earlier handoff.

Table 9 Qwen configuration bound to the inspected snapshot and fixed eight-H20 receipts.

Field	Value
Decoder	64 layers; hidden 5,120; intermediate 17,408
Token mixers	48 GDN; 16 full GQA; 24 query heads, 4 KV heads, head dimension 256
Native position setting	262,144
Adaptation	NF4 QLoRA, rank 16, alpha 32; 116,727,808 trainable parameters
Parallel/storage	8 H20, CP8, page size 64, compact GPU pages
Prompt/response	2,088,960 prompt tokens; 8,192 response input tokens per member; at most 8,191 scored
Replay blocks	510 prompt chunks of 4,096; four response blocks of 2,048; FFN microblock 512
Objective in receipt	Synthetic rewards; old/reference from same runner; $\beta = 0$; live ratio term unclipped
Update boundary	Global response forward; dQ reduced; shard-local K/V adapter gradients not reduced; local AdamW calls

Table 10 GLM configuration bound to the final 2,097,152-position manifest and inspected runtime.

Field	Value
Decoder	78 layers; hidden 6,144; 64 MLA query heads; query/KV latent widths 2,048/512
Sparse attention	32 indexer heads of dimension 128; top-2,048; 21 compute layers and 57 IndexShare consumers
Feed-forward	First 3 dense; remaining 75 MoE; 256 routed experts, top-8, one shared expert; expert intermediate 2,048
Native position setting	1,048,576
Adaptation	LoRA rank 8; attention projection, dense/routed/shared FFN, and output-head target families
Frozen objects	Base weights, embeddings, norms, router parameters, and DSA indexer parameters; router expert-bias transition not audited
Parallel/storage	32 H20; TP1/CP32/EP32/ETP1/PP1; CPU prefix pages; page size 64
Prompt/response	2,097,152 prompt tokens; two three-token response sequences, each yielding two scored next-token positions
Objective in receipt	Rewards $[0, 1]$; advantages $[-1, 1]$; $\epsilon = 0.2$, $\beta = 0.01$; old is reference; ratio one
Update boundary	CP-local DSA; custom replay skips Megatron gradient finalization for CP-replicated non-expert adapters

The final GLM manifest does not bind the actual learning rate, token checksum, LoRA seed, package, driver, and NCCL versions, or final model revision. Earlier handoff documents contain environment snapshots, but they are not part of the canonical receipt and are not promoted into Table 10.

B GLM Page Mapping and State-Size Derivation

Megatron zigzag context parallelism divides the prompt into $2C$ contiguous chunks for $C = 32$. With 32,768 global pages and page size 64, each chunk has 512 pages. Rank r owns chunks r and $63 - r$:

$$\mathcal{P}_r = \{512r, \dots, 512r + 511\} \cup \{512(63 - r), \dots, 512(63 - r) + 511\}. \quad (36)$$

Each rank owns 1,024 pages (65,536 prompt tokens); all 32 ranks have logged endpoint samples.

The page tensor shapes are bound by the live replay contract and a prior shape trace. An MLA page is BF16 $[1, 64, 1, 576]$, where 576 combines the latent and rotary content used by the absorbed path. A DSA indexer-key page is BF16 $[1, 64, 128]$. Materializing a complete local layer produces $[1, 65, 536, 1, 576]$; an index-computing layer also materializes $[1, 65, 536, 128]$.

The following numbers are derived from those shapes, not measured memory peaks. One local MLA layer occupies 72 MiB. One local DSA index-key set occupies 16 MiB. Across 78 MLA states and 21 index-key states, CPU prefix storage is

$$78 \times 72 \text{ MiB} + 21 \times 16 \text{ MiB} = 5,952 \text{ MiB} = 5.8125 \text{ GiB/rank}. \quad (37)$$

The collective CPU payload is 186 GiB. One staged shared-index layer needs a 72 MiB prompt payload before response work; one index-computing layer needs 88 MiB. These figures exclude page metadata, pinned transfer buffers, model weights, response tensors, and allocator overhead.

C Representative GLM Trace Contract

A policy trace begins with response hidden shape $[2, 1, 6144]$. A compute layer records both state components, 1,024 pages, local prefix 65,536, total local length 65,538, top-k shape $[1, 2, 2048]$, CPU offload, whole-layer checkpointing, and backend `runtime_unfused_absorbed`. An IndexShare consumer records MLA state only and consumes the per-forward selection holder published by its producer.

Output logits have shape $[1, 2, 154880]$. Backward runs from layer 77 to 0. Layer and attention-projection gradients are $[2, 1, 6144]$; sparse-attention gradients are $[2, 1, 16384]$. Policy traces contain 396 JSONL events; old traces contain 160. These are host events, not kernel profiles or numerical gradient dumps.

During capture, gradients are disabled for 1,394 parameters and 99 state hooks cover 78 MLA states plus 21 DSA index-key states. One shared RoPE cache replaces 98 repeated copies. Complete decoder layers use reentrant checkpointing with RNG preservation; saved sparse tensors may be moved to CPU.

D Distributed-Gradient Audit

Qwen. The distributed attention backward all-reduces dQ and returns page-owner dK/dV . The latter are correct gradients of sharded KV tensors, but their projection adapters are replicated. The probe creates one AdamW instance per rank without DDP or a parameter-gradient reducer. The required selective K/V and upstream-hidden composition is absent from the audited path.

GLM. The custom resident group loop executes two local backward calls. In the inspected code, these call `loss.backward()` and then `engine.optimizer_step()`. The normal Megatron pipeline schedule calls `finalize_model_grads`, which in turn calls DDP `finish_grad_sync`; the custom loop bypasses this schedule. Gradient overlap is disabled, so backward hooks only accumulate local `main_grad`. The distributed optimizer assumes reduction has already occurred, updates its parameter shard, and all-gathers updated shards. The all-gather can make parameter replicas agree while applying an unreduced local CP gradient.

Routed-expert adapters are not replicas in this topology. EP32 and ETP1 give expert data-parallel size one, and EP all-to-all autograd routes token contributions to each expert owner. The missing finalization is decisive for CP-replicated attention, dense/shared, and output-head adapters. The final trace does not record per-module hook coverage, gradient counts, hashes, or deltas.

E Detailed GLM Capacity Progression

The conventional 2,097,152-position full-sequence attempt exposed several independent peaks. The DSA score scratch [8192, 2,097,152] in FP32 is approximately 64 GiB. After restricting that path, expert-LoRA scale/add work reached 7.80 GiB and an FC2 matmul allocation reached 11.70 GiB. Smaller MoE chunks moved the failure into expert-output concatenation; at chunk size 65,536 the requested concatenate allocation was 19.37 GiB. These failures show why one kernel optimization did not resolve the full graph.

The corresponding capacity milestones were:

1. all-layer no-grad prompt capture at 128K, 256K, 512K, 1M, and 2.097M; the final 2,097,152-position prefix-only capture took 675.657 s;
2. layer-0 2.097M capture, two local backwards, and an optimizer-call canary in 738.579 s;
3. all-layer CP32 gates at 32K and 64K after IndexShare, CPU page, shared RoPE, and checkpoint fixes;
4. a 2.097M $G = 1$ all-78-layer canary in 2042.975 s; and
5. the fresh $G = 2$ rank-complete transaction in 2975.138 s.

The sequence is monotonic in execution coverage, not in semantic fidelity. The same CP-local DSA and skipped gradient-finalization boundaries remain in the last two milestones.

F Qwen 4.25M Replay within Eight H20s

The Qwen investigation also reached a 4.25M context frontier, where 4.25M means $4.25 \times 2^{20} = 4,456,448$ exact positions. The resident response-replay run captures 4,448,256 prompt positions once, then completes all eight old/reference/policy branches for 8,192 response positions. Every policy branch finishes four 2,048-position backward blocks. The group takes 4052.920 s after a 17697.213 s inferred prefix interval, or 21750.133 s from prefix start through $G = 8$, at 82.960 GB per rank. Its old resident command interface intentionally skips the optimizer, so it is a complete replay receipt rather than a step receipt.

Table 11 Qwen 4.25M replay within eight H20s (4,456,448 exact positions). Train-block probes are capacity checks. The resident row completes every old/reference/policy branch and all four backward blocks for each of eight members, but its old command interface skips the optimizer. The prefix-frozen response-only row completes eight $G = 8$ optimizer steps (64 member replays) while keeping the prefix valid. PASS peaks use PyTorch allocator bytes in decimal GB; the * approximate full-run OOM value is sampled process memory and is not directly comparable. None closes the CP8 adapter-gradient reduction gap.

Context length	Run type	Result	Reported GB
4,194,304	train-block proxy	PASS	136.717
4,456,448	train-block proxy	PASS	143.163
4,538,368	train-block proxy	PASS	145.176
4,542,464	train-block proxy	OOM	≈ 145.277
4,456,448	unpruned full run	OOM in policy backward	$\approx 142.293^*$
4,456,448	resident response replay, $G = 8$	PASS, all 8 members	82.960
4,456,448	prefix-frozen response-only, $G = 8$	PASS, 8 optimizer steps	83.894
4,456,448	batched old/reference, $G = 8$	PASS, small speedup	135.128

A separate prefix-frozen response-only run supplies the multi-step evidence. It completes eight $G = 8$ accumulation cycles and eight optimizer steps: 64 member replays in total. Every rank records `optimizer_step_applied=true`, `prefix_stale=false`, and `prefix_frozen_response_only=true`; peak allocation rises from 82.960 GB on the first cycle to 83.894 GB thereafter. The prefix remains valid because prompt-position

adapter deltas are disabled by this parameterization. This is continuous training evidence for that explicit objective, not for the original prompt-adapted QLoRA objective.

Before detached-prefix gradient-page pruning, a clean prompt-adapted run completed prefix capture, old/reference scoring, and policy stage-1 forward, then OOMed in response backward. The pruning is exact for LongStraw’s detached-prefix objective because prompt K/V pages still participate in attention and dQ , while their own unused dK/dV storage is omitted from the live graph.

The linear storage slope follows directly from the Qwen model structure. For 16 full-attention layers with four KV heads of dimension 256 in BF16, sharded over CP8, one million additional global context positions add $10^6 \times 16 \times 2 \times 4 \times 256 \times 2/8 = 8.192$ decimal GB of KV storage per rank. This arithmetic explains the above-4M capacity potential, but it is not a whole-step headroom claim: replay scratch, score buffers, and autograd state still determine whether the complete path fits.

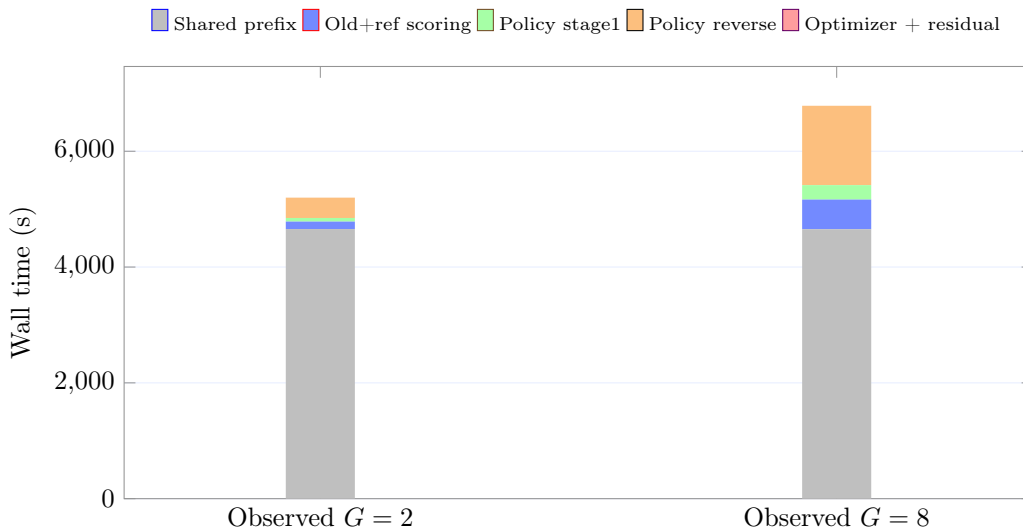


Figure 9 Qwen group accounting within eight H20s. The $G = 2$ and $G = 8$ bars are serial-loop anchors, not group-size limits. Four times as many members add 1,586.448 s but only 0.208 GB (0.213%) at peak because the shared prefix dominates residency; the final segment is a phase-sum residual, not a pure optimizer timer.

Batched old/reference scoring is a useful negative result. At 4K it reduces a one-GPU $G = 8$ total from 17.529 to 14.528 seconds. At 4,456,448 context it changes the post-prefix time from 4051.240 to 4024.290 seconds, only 0.7%, while peak memory rises from 83.894 to 135.128 GB. Window/sparse attention, a 4,096-token response block, metadata batching, an alternative GDN backend, and a naive query split remain diagnostic or rejected variants rather than primary evidence in this report.

Figure 10 separates prefix-only capacity, train-block diagnostics, and complete replay or optimizer-step receipts. Its line connects only the local train-block bracket; it is not a fitted memory law. The low-memory 4.25M points use the detached-prefix objectives described above, whereas the high-memory points retain the prompt-adapted block path.

Read vertically, the prefix-only and conditional-response points establish different execution scopes; their lower memory does not imply that a prompt-adapted training graph fits. The triangle sequence is a train-block diagnostic whose final pass/OOM pair differs by one 4,096-token chunk. The replay diamonds represent completed suffix work, while the conditional point records a response path, not an optimizer-step receipt.

No line connects the completed replay points because there is no matched sweep over context length, objective, and measurement window. The allocator readings are also not directly comparable when their collection windows differ; Table 11 remains the scope key for every plotted point.

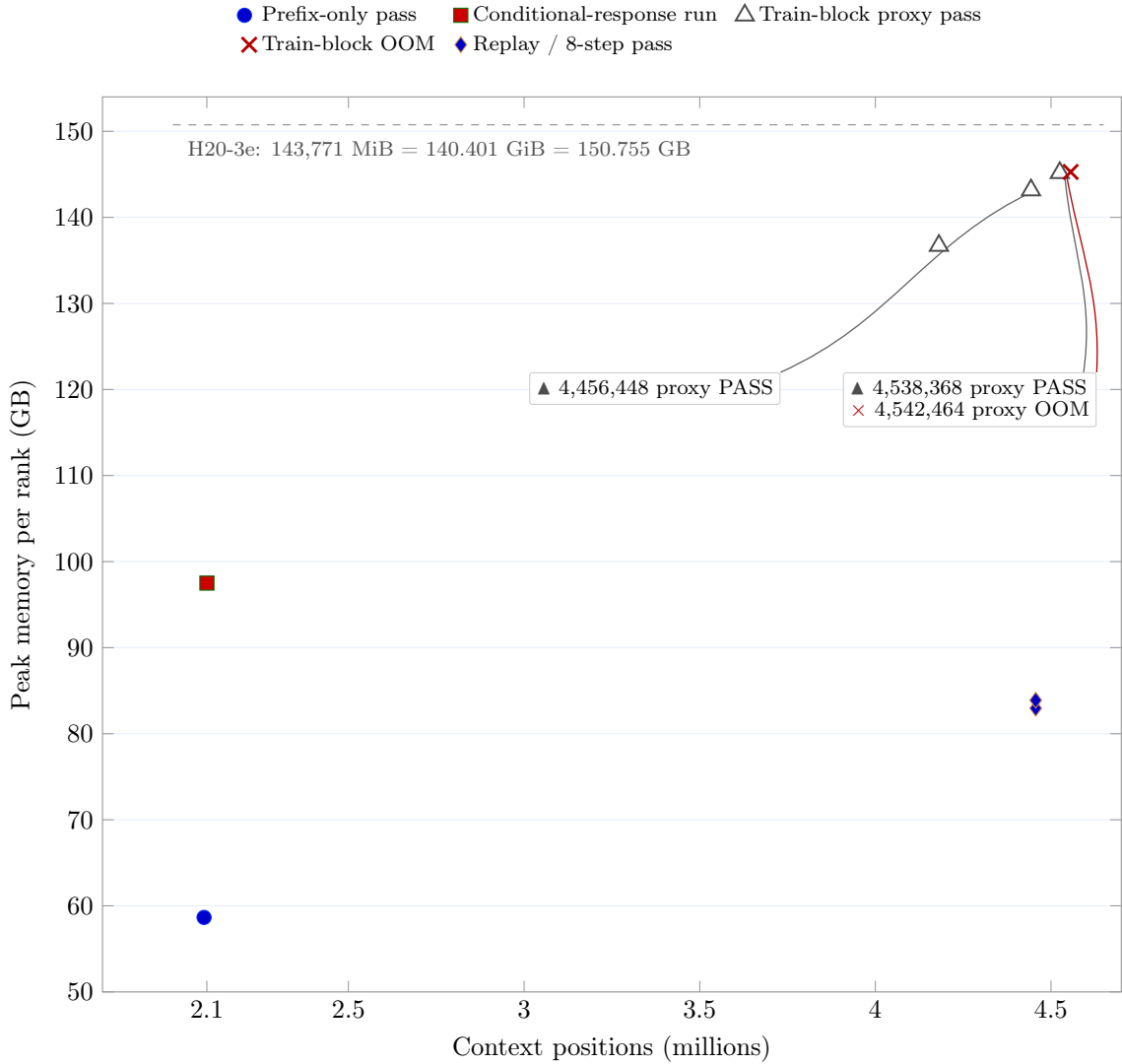


Figure 10 Qwen 4.25M replay within eight H20s. Here 4.25M denotes 4,456,448 exact positions. At this scale, the resident $G = 8$ path completes all response replays at 82.960 GB, and the prefix-frozen response-only variant completes eight optimizer steps at 83.894 GB. A train-block proxy reaches 4,538,368 and OOMs one 4,096-position chunk later at 4,542,464. The clean prompt-adapted path before detached-prefix gradient-page pruning OOMs in policy backward at 4,456,448; it remains in Table 11, but its sampled process-memory reading is omitted here because it is not comparable with allocator peaks. The 8-step result is complete for the explicit prefix-frozen objective; none of these points closes the independent CP8 adapter-gradient reduction gap.

G Acknowledgements

We thank the following MindLab members for their support and contributions to the broader research environment: Theo Li, Song Cao, Wenbin Wang, Fancy Kong, Regis Ye, Charles Huang, Murphy Zhuang, Josh Ying, Anya Zhang, Alyssa, Ray Li, Logan Liu, Xiang Liu, Yuhao Zhan, Kaixuan Fan, Mutian Hong, Zhuoran Shen, Hua Jiang, Wenxi Qu, Yuxin Lu, Neo Liu, Hera Feng, Aaron Guan, Fan Lin, Guoshuai Han, Xinyue Zhu, Chengdong Xu, Jingwei Cao, Smith Li, Kun Li, Jianbo Wu, Yuyi Jiang, Sueky Zhang, Kairus Liu, Zhihui Li, Wei Zhao, Anson Qiu, Hongquan Gu, Peixuan Hua, Nora Jiang, Ada Zhou, Qiuyu Jin, Ruijia Zhang, Arthur Fu, Maxwell Yao, Jiayi Lin, Runze Lv, Hailee Hou, Miles Jiang, Ya Zhang, Danney Zeng, Vin Bo, and Jason Zhang. We also thank the NVIDIA team for its support.