Strata Memory: Stratified Compressed Context for Long-Context Language Models

Anonymous ACL submission

Abstract

Activation-level context compression reduces inference cost in long-context applications by replacing historical tokens with compact carrier states. However, compressed histories must support both sparse long-range structural coverage and precise local-evidence recovery, while existing methods typically keep carrier states in a homogeneous stream with one access interface. We identify this mismatch as role conflict and introduce Strata Memory, a role-structured framework for activation-level context compression under a fixed carrier-write budget. Strata Memory assigns inserted carriers to complementary Anchor and Detail memories, aligning sparse long-range access with denser local-evidence access. It structures only carrier states, without partitioning raw tokens, changing the training objective, or increasing written carriers. Under protocols matched to the closest homogeneous activation-compression baseline, Strata Memory improves LongBench performance across Llama and Qwen backbones, with gains up to 3.7 points on Multi-Doc and 2.7 points on Code. Ablations show that gains arise from aligning write-time carrier roles with read-time access policies, rather than from typed carriers, bank-specific access, or dual banks alone. Strata Memory also degrades more slowly up to 32× compression and adds negligible measured latency and peak-memory overhead, showing that compressed-state organization and access matter beyond carrier-state count.

1 Introduction

Large language models are moving from isolated prompt answering toward tasks that require maintaining long, evolving histories. Long-document analysis and multi-document question answering require retaining dispersed evidence across sources (Bai et al., 2024; Shaham et al., 2022; Yang et al., 2018; Ho et al., 2020); repository-level code reasoning and extended dialogue require tracking definitions, decisions, and prior turns over long ranges (Liu et al., 2024; Zhong et al., 2021). Tool-using and agentic workflows further lengthen the operational history a model must maintain, making full-history attention increasingly costly in computation and key-value memory.

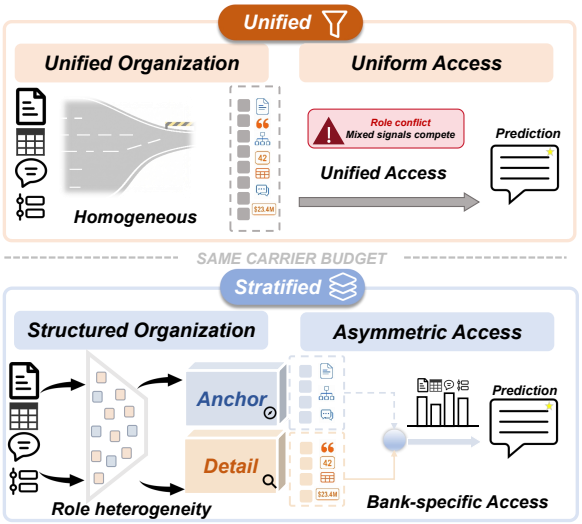

Figure 1: Homogeneous compressed memory versus Strata Memory. Both use the same written carrier-state budget. Homogeneous compression stores carrier states in a single stream with unified access, whereas Strata Memory writes them into Anchor and Detail banks and reads them with role-specific access.

Context compression addresses this cost by replacing raw histories with compact states. Recurrent and compressive memories reuse information from previous segments through explicit memory states (Rae et al., 2020; Bulatov et al., 2022). Other approaches learn soft summary states or memory slots that later tokens can condition on (Chevalier et al., 2023; Ge et al., 2024). Inference-time methods instead reduce the visible context through prompt compression or cache selection (Jiang et al., 2024; Pan et al., 2024; Li et al., 2024). More recent activation-level methods insert carrier tokens and retain their hidden states as continuous memories

(Zhang et al., 2025). These methods mainly study how compressed states are produced and trained.

However, the histories being compressed are not functionally homogeneous. Some signals provide long-range organization, such as topic transitions, entity continuity, and cross-file dependencies. Others provide local evidence, such as quoted spans, numerical values, and function definitions. Effective compressed memory must therefore support both broad coverage and fine-grained evidence recovery under a limited carrier budget. This raises a complementary question, illustrated in Figure 1: after history has been compressed into carrier states, should all retained states still be organized and read through the same interface?

Homogeneous compressed memory stores all retained carrier states in a single stream and exposes them through one access rule. This interface is efficient, but it makes states with different predictive roles functionally interchangeable: a state useful for long-range structure is read in the same way as a state needed for local evidence recovery. We refer to this mismatch as role conflict. The issue is not only how many states are retained, but also whether the retained states have an interface that matches their use during prediction.

We propose Strata Memory, a role-structured context-compression framework that organizes retained carrier states without increasing written carrier states. Inserted carriers are written into complementary banks: Anchor Memory targets sparse long-range coverage, while Detail Memory uses a denser read policy for local evidence. The roles are assigned to carrier states rather than raw input tokens, so Strata Memory does not partition the input, add retrieval supervision, or increase the carrier-write budget. Its central principle is write–read consistency: carriers are assigned roles at write time and later accessed through policies aligned with those roles. This design should be most useful when tasks require both cross-context aggregation and local evidence recovery, and when the carrier-write budget becomes tighter.

We evaluate these expectations under matched carrier budgets and compression protocols, using homogeneous carrier memory as the main controlled baseline. The evaluation is designed to separate role-structured organization from larger memory budgets and from isolated component changes. It tests performance, compression scaling, and write–read alignment under matched protocols. Appendix A further verifies that the role-structured interface adds negligible overhead relative to the homogeneous carrier baseline.

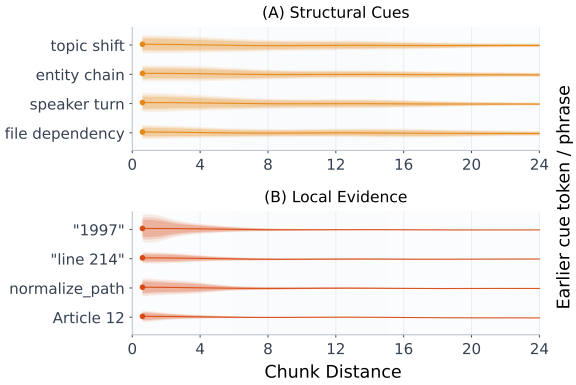

Figure 2: Measured cue-range diagnostic motivating role-structured memory. Representative structural cues show broader future use, whereas local-evidence cues are more localized. Shaded regions indicate variation across measured cue instances.

In summary, our core contributions are:

- We propose Strata Memory, a fixed-budget post-compression framework for organizing activation-level carrier states without raw-token partitioning, retrieval supervision, or additional carrier budget.
- We design a role-consistent write–read interface that couples typed carrier writing with bank-aligned access through Anchor and Detail memory banks.
- We show through controlled experiments that Strata Memory improves compression quality, remains robust under tighter budgets, and preserves the efficiency profile of homogeneous activation compression.

2 Related Work

Efficient context representations. Context compression reduces long-context cost by replacing raw histories with compact states. Recurrent and compressive models reuse segment-level memory states across context blocks (Rae et al., 2020; Bulatov et al., 2022), while soft-compression methods train summary tokens or memory slots that later tokens condition on (Chevalier et al., 2023; Ge et al., 2024). Activation-level compression further retains carrier-token activations as layer-wise key-value memories (Zhang et al., 2025). A related line reduces the visible context directly: prompt-compression methods remove or rewrite

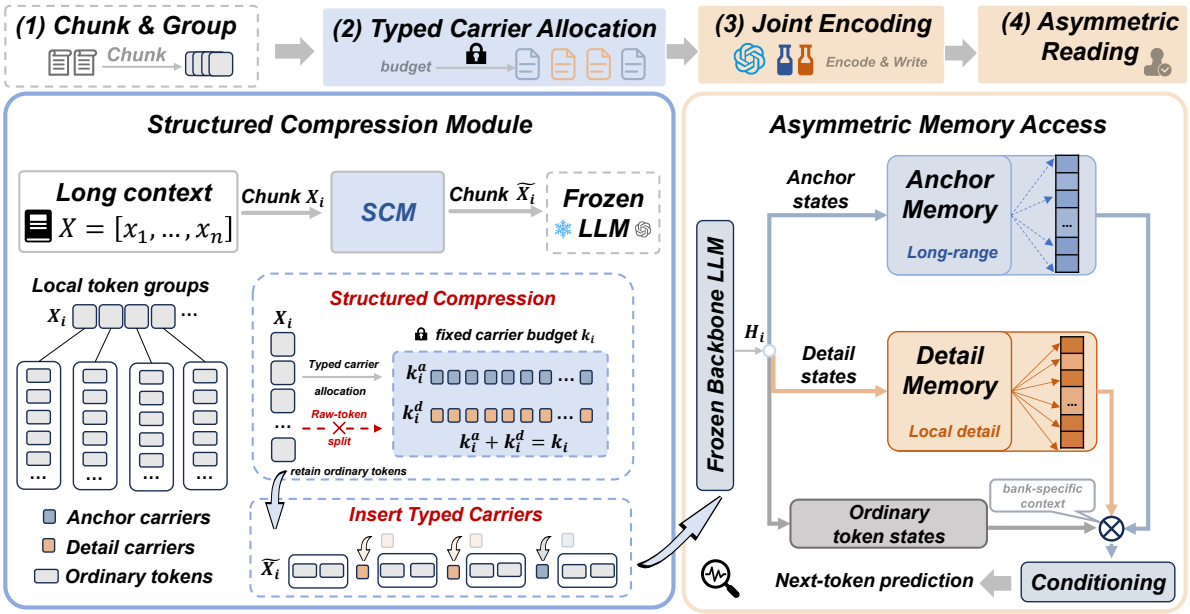

Figure 3: Overview of **Strata Memory**. Given a fixed written carrier-state budget, Strata Memory inserts typed carriers into each causal chunk and writes their encoded states into Anchor Memory and Detail Memory. The two banks are accessed with bank-consistent policies for sparse long-range coverage and denser access to local evidence.

less useful input tokens before inference (Jiang et al., 2024; Pan et al., 2024), KV-cache methods prune, merge, allocate, or sparsify token-level cached states (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024; Sun et al., 2025), and streaming attention restricts decoding visibility to sink and recent tokens (Xiao et al., 2024). These approaches focus on producing compact states or selecting which raw-token states remain visible. Strata Memory addresses a complementary question: given a fixed written carrier-state budget, how should activation-compressed carrier states be organized and read?

Structured memory. Structured memory has been explored through hierarchical segment memories, recurrent aggregation, and persistent latent modules (Wang et al., 2023; He et al., 2025; Chen et al., 2025; Wang et al., 2025). Sparse-attention architectures also combine broad and local context through global-local attention patterns (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). Strata Memory differs from these designs in that it neither partitions raw tokens into global/local subsets nor introduces an external retrieval module or additional memory budget. Instead, it structures the carrier states produced by activation-level compression itself, aligning their write-time roles with bank-specific read policies.

3 Method

Strata Memory keeps the activation-compression pipeline and written carrier-state budget fixed, but changes the memory interface after carrier states are produced. Here, carrier budget means states written during compression; query-visible memory is determined separately by the read policies. The method assigns roles before encoding, writes carrier states into role-specific banks, and reads each bank with a role-matched policy. Figure 3 gives an overview, and Table 2 later isolates typed writing, bank-specific reading, uniform access, and mismatched access.

3.1 Problem Setup

We first fix the base compression setting shared by all variants. The carrier budget and causal encoding follow activation-level compression; Strata Memory changes only how the retained carrier states are organized and read. Figure 2 shows a measured cue-range diagnostic motivating this design choice. Representative structural cues exhibit broader future-use ranges than local-evidence cues, and the labels are used only for analysis, not for training or role assignment.

Given a long sequence $X = (x_1, \ldots, x_n)$, we split it into causal chunks $X_1, \ldots, X_T$, each with length at most w. Following activation-level com-

pression (Zhang et al., 2025), each chunk is augmented with inserted carrier positions whose layer-wise key-value states can be retained after encoding. For a target compression factor α, chunk X_i is divided into local groups $G_{i,1}, \ldots, G_{i,k_i}$, where

$$k_i = \left\lceil \frac{|X_i|}{\alpha} \right\rceil. \tag{1}$$

The augmented chunk is

$$\widetilde{X}_i = [G_{i,1}; c_{i,1}; \ldots; G_{i,k_i}; c_{i,k_i}], \tag{2}$$

where $c_{i,j}$ denotes the j-th inserted carrier position. The carrier-write budget is fixed by k_i. Strata Memory does not increase this written-state budget; it only changes how the resulting carrier states are typed, stored, and read.

Training uses standard next-token prediction over ordinary tokens in the original sequence. Inserted carriers are auxiliary memory positions and are masked out of the loss; labels follow the original token order rather than the carrier-augmented sequence. The model learns role-specific carrier behavior without manual role supervision.

3.2 Role-Structured Carrier Allocation

We next define the write-time roles of inserted carriers. Before encoding a chunk, Strata Memory divides the existing carrier positions into Anchor and Detail roles without changing their total number. Let $\rho \in (0, 1)$ be the anchor-budget ratio. For chunks with $k_i > 1$, the number of Anchor carriers is

$$k_i^a = \min\{k_i - 1, \max(1, \mathrm{round}(\rho k_i))\}. \tag{3}$$

The remaining $k_i^d = k_i - k_i^a$ carriers are assigned to Detail Memory, so the total carrier budget remains k_i. For the degenerate case $k_i = 1$, the only carrier is assigned to Detail Memory. This allocation separates coarse temporal coverage from local resolution under a fixed carrier-write budget. Uniformly placed Anchor carriers provide a compressed historical scaffold, while Detail carriers keep denser carrier-state coverage around local spans.

Anchor carriers are distributed uniformly over the carrier timeline, and all non-Anchor carrier positions are assigned to Detail Memory. We denote the resulting sets by $\mathcal{A}_i$ and $\mathcal{D}_i$, respectively. The exact deterministic placement rule is given in Appendix H. For a carrier position t, its type is $\tau(t) = a$ if $t \in \mathcal{A}_i$ and $\tau(t) = d$ if $t \in \mathcal{D}_i$.

This assignment is not a token-level split of the input. Ordinary tokens remain in their original causal order, and both carrier types are inserted into the same augmented chunk. The role names describe the intended storage and access behavior after encoding: Anchor states are read for sparse long-range coverage, while Detail states use a denser read interface for access to local evidence.

3.3 Type-Aware Carrier Encoding

After roles are assigned, they enter the model only through carrier-token projections. Ordinary tokens keep the frozen backbone projections, while carrier tokens use trainable projections with Anchor/Detail-specific scaling. We use I_i^r and I_i^c to denote ordinary-token positions and carrier positions in $\widetilde{X}_i$. At layer ℓ, let $h_{i,t}^\ell$ be the input hidden state at position t. For projection type $P \in \{Q, K, V\}$:

$$z_{i,t}^{P,\ell} = \begin{cases} W_P^\ell h_{i,t}^\ell, & t \in I_i^r, \\ (W_{P,c}^\ell h_{i,t}^\ell) \odot \eta_{P,\tau(t)}^\ell, & t \in I_i^c. \end{cases} \tag{4}$$

Here W_P^ℓ belongs to the frozen backbone, while $W_{P,c}^\ell$ and $\eta_{P,\tau(t)}^\ell$ are trainable compression parameters. The definition is layer-local: projections are computed from the layer input before the attention output of that layer is produced, so the formulation introduces no circular dependency.

After chunk X_i is encoded, only typed carrier key-value states are appended to the persistent compressed-memory banks. Let $U_i^{m,\ell}$ denote the key-value states at carrier positions of type $m \in \{a, d\}$ in layer ℓ. The bank update is

$$M_i^{m,\ell} = [M_{i-1}^{m,\ell}; U_i^{m,\ell}], \quad m \in \{a, d\}. \tag{5}$$

The Anchor and Detail banks constitute the persistent compressed memory used for historical context.

3.4 Bank-Consistent Reading

We finally define how previously written banks are read by later chunks. Before processing chunk X_i, the model reads historical memory only from $M_{i-1}^{a,\ell}$ and $M_{i-1}^{d,\ell}$; carrier states from X_i are appended after encoding and cannot be read back within the same chunk. Anchor Memory is read sparsely along the memory timeline:

$$S_i^{a,\ell} = \mathrm{Unif}(M_{i-1}^{a,\ell}, T_a), \tag{6}$$

where T_a is the anchor read budget. The operator $\mathrm{Unif}(\cdot, T_a)$ keeps at most T_a uniformly spaced

states; if the bank is shorter than T_a, all Anchor states are retained.

Detail Memory is read with a less sparsified local-evidence-oriented policy:

$$S_i^{d,\ell} = \mathrm{Read}_d(M_{i-1}^{d,\ell}; T_d). \tag{7}$$

Here $\mathrm{Read}_d(M; T_d)$ returns the full bank M when $T_d = \infty$, and otherwise returns the most recent T_d states. Strata Memory keeps the compression pipeline and written carrier budget fixed, but reorganizes carrier storage and access. It assigns carrier roles before encoding, writes them into role-specific banks, and reads each bank accordingly.

At layer ℓ, the memory-side context is the ordered concatenation of the two compressed banks:

$$R_i^{\ell} = [S_i^{a,\ell}; S_i^{d,\ell}]. \tag{8}$$

For a query position t in the current augmented chunk, let $\mathcal{C}_i^{\ell}(t)$ denote the key-value states in $\widetilde{X}_i$ whose positions are no later than t. The visible context is

$$\Omega_i^{\ell}(t) = [R_i^{\ell}; \mathcal{C}_i^{\ell}(t)]. \tag{9}$$

The attention mask admits exactly the states in $\Omega_i^{\ell}(t)$ and suppresses all others. Thus, persistent memory used as historical context comes only from previous chunks, while the current chunk is processed causally. Ordinary tokens may attend to earlier carriers in the same augmented chunk, but those carriers summarize only earlier causal prefixes. This prevents both self-referential memory updates and future-token leakage.

The persistent historical memory consists only of compressed Anchor and Detail carrier states. The carrier budget determines how many compressed states are written and retained, while the read policies determine how many retained states are visible to each query. Detailed allocation, read-cost definitions, and visible-state growth analysis are provided in Appendices H and I.

4 Experiments

4.1 Experimental Setup

Models. We evaluate Strata Memory with frozen backbone parameters and train only memory-specific components, including carrier embeddings, carrier-specific projections, and type-aware scaling parameters. The main backbones are Llama-2-7B-Chat (Touvron et al., 2023) and Qwen-2-7B-Instruct (Yang et al., 2024); we additionally evaluate Qwen-2.5-7B and Qwen-2.5-14B to test robustness across model versions and scales. The chunk size is $w = 1024$ for Llama-2 and $w = 2048$ for Qwen-family models. Unless otherwise specified, Strata Memory uses $\rho = 0.5$, $T_a = 256$, and $T_d = \infty$. This default caps Anchor reads but keeps Detail reads visible over compressed carrier states; sensitivity analyses are reported in Appendices G and K, and visible-state growth is analyzed in Appendix I.

Compression protocol. Reported input length always refers to the original context length before compression. For LongBench, we follow the corresponding activation-compression protocol: Llama-2 uses adaptive $2\times/4\times/8\times$ compression for 4K–8K, 8K–16K, and 16K–32K inputs, respectively, while Qwen-family models use uniform $4\times$ compression. Unless otherwise specified, each benchmark result is obtained from a single deterministic evaluation of the final trained checkpoint. All methods within each model block use the same prompts, generation limits, stopping rules, decoding configuration, and, where applicable, random seed. LongBench category scores are unweighted means over datasets within each category, with per-dataset results in Appendix D. Efficiency experiments are reported separately under the $8\times$ setting.

Training. Training follows a two-stage memory adaptation recipe. The first stage uses 1B tokens sampled from RedPajama (Weber et al., 2024); the second stage uses LongAlpaca (Chen et al., 2024), BookSum (Kryscinski et al., 2022), and synthetic long-context instruction data. All training samples are shorter than 20K tokens. We use bf16 precision, batch size 8, linear learning-rate decay without warmup, and learning rates of 5×10^{-5} and 1×10^{-5} for the two stages. Training is conducted on $4\times$ NVIDIA A100 GPUs.

Baselines. We compare with full-attention references and compressed-context baselines. Full denotes the original backbone with full attention within its available context window, and Full-FT denotes a full-attention reference trained with the same data but not parameter-matched to our frozen-backbone setting. Compressed baselines include AutoCompressors (Chevalier et al., 2023), ICAE (Ge et al., 2024), LongLLMLingua (Jiang et al., 2024), SnapKV (Li et al., 2024), and Activation Beacon (Zhang et al., 2025). Activation Beacon is the closest controlled baseline because it also stores activation-level carrier states; we match its carrier budget, compression protocol, original

Model	Method	Length	Single-Doc	Multi-Doc	Summ.	Few-Shot	Code
Llama-2-7B	Full	4K	24.7	22.4	24.6	**63.2**	57.7
	Full-FT	32K	34.8	27.5	23.2	61.8	**58.5**
	AutoCompr.	32K	12.9	16.4	16.3	23.8	39.4
	ICAE	32K	19.5	19.2	19.5	24.8	27.8
	LongLLML.	32K	21.5	18.8	21.7	49.5	53.2
	SnapKV	4K	24.2	22.6	16.3	60.1	57.7
	Act. Beacon	32K	34.9	27.5	25.0	61.4	57.8
	Ours	32K	**36.1**	**27.9**	**26.3**	61.7	**58.5**
Qwen-2-7B	Full	32K	38.8	37.5	26.7	**70.1**	60.3
	Full-FT	32K	41.0	40.6	26.8	68.5	66.1
	LongLLML.	32K	24.7	20.3	26.3	55.9	50.1
	SnapKV	32K	38.7	37.6	26.2	67.1	60.3
	Act. Beacon	32K	40.5	40.3	26.8	68.4	66.4
	Ours	32K	**42.2**	**43.2**	**27.3**	69.1	**68.5**
Qwen-2.5-7B	Full	32K	41.9	45.2	26.5	69.1	64.9
	Full-FT	32K	42.7	46.1	26.7	67.6	66.3
	Act. Beacon	32K	42.5	45.8	26.8	67.4	66.4
	Ours	32K	**43.9**	**47.4**	**27.2**	**69.5**	**69.1**
Qwen-2.5-14B	Full	32K	42.5	52.9	25.1	**71.7**	66.7
	Full-FT	32K	43.9	50.5	27.1	68.8	67.1
	Act. Beacon	32K	43.4	49.9	27.1	68.5	67.4
	Ours	32K	**44.8**	**53.6**	**27.9**	71.5	**69.2**

Table 1: Long-context benchmark results by category. Length denotes the original input context length before compression. Strata Memory and Activation Beacon use matched written carrier-state budgets and compression protocols within each model block. Full and Full-FT are uncompressed full-attention references. Bold indicates the best result within each model block.

input length, and decoding configuration. For baselines without support for a given length or backbone, we report the strongest applicable setting from the corresponding protocol. Additional implementation details and short-context sanity checks are provided in Appendices L and B.

4.2 Compressed-Context Comparison

Table 1 evaluates whether reorganizing compressed carrier states improves long-context prediction under the same compression protocol. The controlled comparison is against Activation Beacon, which uses the same activation-level compression setting but retains carrier states as a homogeneous memory stream. This comparison isolates the effect of retained-state organization and access under a matched carrier budget.

Strata Memory consistently improves over Activation Beacon across the four backbone settings. The gains are most visible on Multi-Doc and Code, two categories that naturally combine cross-segment aggregation with local evidence recovery. For example, on Qwen-2.5-14B, Multi-Doc improves from 49.9 to 53.6, and Code improves from 67.4 to 69.2. This pattern is consistent with the intended division between sparse historical coverage and local-evidence-oriented access.

The full-attention rows are included as reference points rather than parameter-matched baselines. On Llama-2-7B, the Full row is limited to the native 4K context window, whereas compressed-context methods operate on the original 32K inputs after compression. On Qwen-family models, Strata Memory remains competitive with the full-attention references while keeping the backbone frozen. The primary controlled comparison remains the compressed baseline with the same written carrier-state budget. The per-dataset breakdown in Appendix D further shows that the gain is broad rather than driven by a small number of outlier tasks: on Qwen-2-7B, Strata Memory improves over Activation Beacon on all fourteen LongBench datasets, with the largest gains on Multi-Doc QA and Code. Efficiency measurements against both

Method	Single-Doc	Multi-Doc	Summ.	Few-Shot	Code
No Role Structure (same budget)	40.4	40.2	26.7	68.3	66.3
Typed Carriers Only	40.8	40.9	26.9	68.6	66.8
Bank-Specific Access Only	40.6	40.7	26.8	68.5	66.7
Dual Banks + Uniform Access	40.7	41.0	26.9	68.6	66.9
Swapped Access Policies	39.2	39.7	26.1	67.5	65.9
Strata Memory (Ours)	**42.2**	**43.2**	**27.3**	**69.1**	**68.5**

Table 2: Ablation of write–read consistency on Qwen-2-7B. All variants use the same written carrier-state budget and evaluation protocol. The no-role variant collapses carriers into one untyped memory stream, while the other rows isolate typed writing, bank-specific reading, uniform dual-bank access, and mismatched access policies.

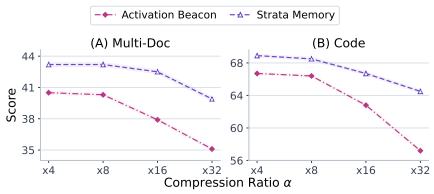

Figure 4: Compression-ratio scaling on representative LongBench categories. Scores are reported under compression ratios $\alpha \in \{4, 8, 16, 32\}$.

full-context inference and Activation Beacon are reported in Appendix A.

4.3 Compression-Ratio Scaling

Figure 4 tests whether the benefit of role-structured memory becomes more important when the carrier budget is tightened. We focus on Multi-Doc and Code because these categories require both locating relevant long-range context and recovering local evidence from the selected region. As the compression ratio increases, each retained carrier must summarize a larger span of history, making a homogeneous carrier stream more likely to mix structural cues and local evidence.

Strata Memory shows a more stable degradation pattern under aggressive compression. This does not come from writing more compressed states: the written carrier-state budget is matched at each compression ratio. Instead, the result suggests that preserving separate write–read roles helps the compressed memory remain usable when fewer carrier states are available. Full scaling results over all five LongBench categories are provided in Appendix C.

4.4 Ablation Study

Table 2 isolates whether the improvement of Strata Memory comes from role-consistent memory organization rather than from retaining more compressed states or simply adding separate components. The no-role variant serves as an internal homogeneous-memory sanity check. It preserves the same written carrier-state budget, training recipe, compression protocol, and decoding configuration, but collapses all carriers into one untyped memory stream with unified access. This mirrors the homogeneous carrier-memory interface used by the closest activation-compression baseline, while keeping the implementation path controlled. Its competitive performance is important: the comparison does not rely on weakening the same-budget baseline. Appendix J further shows that Strata Memory consistently outperforms the homogeneous no-role variant under matched visible-state budgets, indicating that the gain is not due to a larger read interface.

The intermediate variants test different ways of breaking write–read consistency. Typed carriers alone add role information at write time, but the resulting states are still consumed through a unified read rule. Bank-specific access alone changes the read policy, but the carriers are not written with distinct roles. Dual banks with uniform access test whether separating storage is sufficient without role-specific reading. All three variants remain below the complete model, indicating that neither typed writing, asymmetric reading, nor two storage pools alone explains the result.

The swapped-access variant provides the strongest negative control. It keeps the two banks but mismatches the read policy with the write-time role, causing performance to drop across categories. This supports the central claim that Strata Memory is a write–read consistency mechanism rather than an additive combination of independent tricks. Formal definitions of all ablation variants are provided in Appendix E. Additional single-role ablations, anchor-ratio sensitivity, and read-budget sensitivity are reported in Appendices F, G, and K.

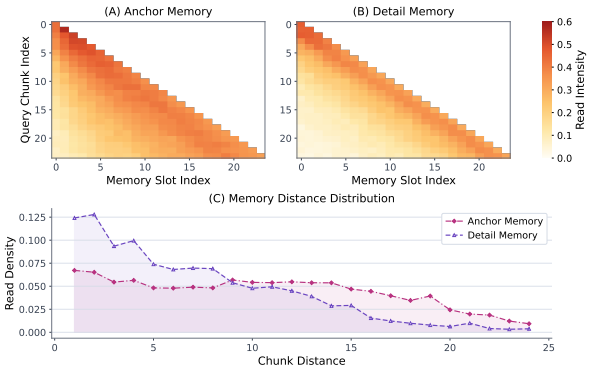

Figure 5: Distance-aware memory access patterns measured by normalized attention mass during prediction. Anchor Memory exhibits broader long-range access, while Detail Memory concentrates more mass on nearby chunks.

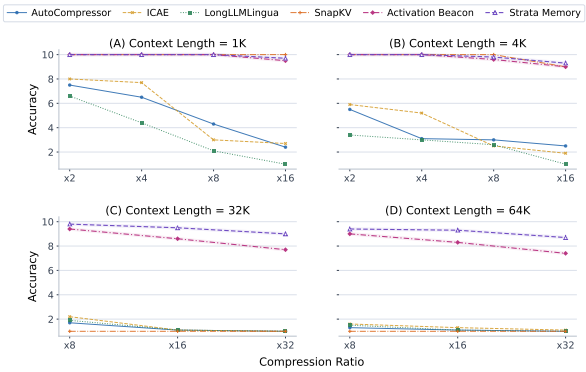

Figure 6: Needle-in-a-Haystack evaluation on Llama-2 across effective context lengths and compression ratios. Strata Memory better preserves sparse inserted evidence in compressed contexts, especially under longer contexts and more aggressive compression.

4.5 Memory Access Analysis

Figure 5 examines how the two banks are used during prediction. This analysis is meant to complement the ablation study: Table 2 shows that aligning write-time roles with read-time policies improves task performance, while Figure 5 checks whether the resulting banks are accessed differently in practice.

Anchor Memory receives broader attention over distant memory slots within historical memory, whereas Detail Memory concentrates more mass on nearby chunks. Under the bank-specific read policies, this pattern is consistent with the intended division between sparse historical coverage and local-evidence-oriented access in the model. The separation is not imposed by an external retrieval module at query time; rather, it reflects learned attention behavior under typed carrier encoding and bank-consistent reading. Together with the ablation results, this provides behavioral evidence that write–read roles affect task performance and compressed-state use during prediction.

4.6 Compression Robustness

We further evaluate sparse evidence recovery with the Needle-in-a-Haystack task. Unlike LongBench categories, this setting asks whether a compressed memory can preserve a small prediction-critical fact inside a long distractor context. Homogeneous compression can dilute such evidence when many heterogeneous signals share the same compressed stream.

Figure 6 shows that Strata Memory remains more stable as the effective context length and compression ratio increase. This trend matches the compression-ratio analysis in Figure 4: role-structured memory is most useful when each retained carrier must summarize a larger portion of history. Together, the benchmark and retrieval-style analyses indicate stronger robustness in sparse evidence recovery under constrained memory budgets across context lengths.

The robustness results also suggest transfer beyond the memory-adaptation lengths used during training. Training samples are shorter than 20K tokens, whereas the main LongBench evaluation uses 32K original contexts and the Needle-in-a-Haystack setting evaluates longer effective contexts. The 128K measurements in Appendix A focus on inference efficiency.

5 Conclusion

We introduced Strata Memory, a role-structured framework that organizes compressed carrier states instead of treating them as a homogeneous memory stream. Without increasing the number of inserted or retained carrier states, Strata Memory writes carrier states into Anchor and Detail banks and reads them with bank-consistent policies for sparse long-range coverage and local-evidence-oriented access. Across Llama and Qwen backbones, the method improves over the closest activation-compression baseline under matched protocols and the same carrier budget while preserving compressed-inference efficiency. The results show that long-context compression depends not only on how many states are retained, but also on how compressed memories are organized and read.

Limitations

This work has several limitations. First, Strata Memory uses two predefined memory roles and a deterministic role-allocation scheme. While this design keeps the method simple and controlled, adaptive role assignment may be more suitable for heterogeneous domains and task-dependent contexts. Second, the default read policy keeps Detail Memory fully visible. As a result, the visible memory length is not constant; it grows over compressed carrier states rather than raw-token history. Although we analyze this behavior and provide matched visible-state comparisons, more tightly bounded read policies may be needed for deployment. Finally, our experiments focus on decoder-only language models and standard long-context benchmarks. Broader evaluation on multilingual, multimodal, and agentic settings remains future work.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069.

Yinpeng Chen, DeLesley Hutchins, Aren Jansen, Andrey Zhmoginov, David Racz, and Jesper Sparre Andersen. 2025. MELODI: exploring memory compression for long contexts. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics.

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

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zifan He, Yingqi Cao, Zongyue Qin, Neha Prakriya, Yizhou Sun, and Jason Cong. 2025. HMT: hierarchical memory transformer for efficient long context language processing. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 8068–8089. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language

understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1658–1677. Association for Computational Linguistics.

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Findings of ACL, pages 6536–6558. Association for Computational Linguistics.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Tianyang Liu, Canwen Xu, and Julian J. McAuley. 2024. Repobench: Benchmarking repository-level code auto-completion systems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pages 963–981. Association for Computational Linguistics.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: standardized comparison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 12007–12021. Association for Computational Linguistics.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. 2025. Shadowkv: KV cache in shadows for high-throughput long-context LLM inference. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian J. McAuley, Dan Gutfreund, Rogério Feris, and Zexue He. 2025. M+: extending memoryllm with scalable long-term memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan

Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *CoRR*, abs/2407.10671.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2025. Long context compression with activation beacon. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics.

A Efficiency Analysis

Inference efficiency is a practical benefit of activation-level context compression. During decoding, a long-context model repeatedly reads accumulated key-value states, so reducing the effective memory length can lower both latency and peak GPU memory usage. To avoid hardware-related confounds, we re-evaluate Full, Activation Beacon, and Strata Memory on the same $4\times$A100 platform under matched decoding settings.

Table 3 reports the prefill latency, decoding latency, total latency, and peak GPU memory under the $8\times$ compression setting. The full-context rows serve as uncompressed references. Compared with full-context inference, both compressed methods substantially reduce decoding latency and GPU memory usage. For example, on Qwen-2.5-7B at 128K input length, total latency decreases from 5.546s for Full to 2.587s for Activation Beacon and 2.594s for Strata Memory, while peak GPU memory drops from 64.7GB to 24.7GB and 24.9GB, respectively. On Qwen-2.5-14B, full-context inference runs out of memory at 128K, whereas both compressed variants remain executable at about 40GB peak memory.

Strata Memory also preserves the efficiency profile of Activation Beacon. Across the four matched settings in Table 3, Strata Memory differs from Activation Beacon by at most 0.008s in total latency and 0.2GB in peak GPU memory under our measurement protocol. Thus, the role-structured organization preserves the practical efficiency profile of the homogeneous activation-compression baseline, while the main text shows that it improves compression quality.

B Short-Context Capabilities

We evaluate whether Strata Memory preserves the backbone model's short-context capabilities after long-context memory adaptation. This is a sanity check rather than a long-context compression benchmark. We report results on MMLU (Hendrycks et al., 2021), ARC-Challenge (Bhakthavatsalam et al., 2021), BoolQ (Clark et al., 2019), and GSM8K (Cobbe et al., 2021). The backbone parameters are kept frozen, and only memory-specific parameters are trained.

Table 4 shows that Strata Memory largely preserves short-context capabilities. Compared with the original backbone, Strata Memory changes MMLU by at most 0.9 points and leaves ARC-C, BoolQ, and GSM8K nearly unchanged. Its short-context behavior is also comparable to Activation Beacon. These results suggest that the long-context gains of Strata Memory are not obtained by substantially degrading the backbone's standard short-context abilities.

Model	Input Length	Method	Prefill	Decode	Total	GPU Memory
Qwen-2.5-7B	32K	Full	0.536	0.726	1.262	27.3
		Activation Beacon	0.515	0.507	1.022	18.2
		Strata Memory	0.516	0.508	1.024	18.0
	128K	Full	3.428	2.118	5.546	64.7
		Activation Beacon	2.036	0.551	2.587	24.7
		Strata Memory	2.035	0.559	2.594	24.9
Qwen-2.5-14B	32K	Full	0.981	1.889	2.870	46.0
		Activation Beacon	0.997	0.856	1.853	36.0
		Strata Memory	0.998	0.858	1.856	36.1
	128K	Full	–	–	–	OOM
		Activation Beacon	4.338	1.116	5.454	40.2
		Strata Memory	4.342	1.120	5.462	40.4

Table 3: Inference efficiency under the $8\times$ compression setting. Latency is reported in seconds, and GPU memory is reported in GB. All three methods are measured on the same $4\times$A100 platform with matched decoding settings. Full denotes full-context inference without compression.

Model	Method	MMLU	ARC-C	BoolQ	GSM8K
Llama-2-7B	Full	47.5	48.5	86.2	9.2
	Act. Beacon	46.6	48.4	86.5	9.3
	Strata Memory	46.9	48.4	86.4	9.4
Qwen-2-7B	Full	70.1	62.7	87.1	76.0
	Act. Beacon	69.1	62.7	87.2	76.2
	Strata Memory	69.2	62.8	87.4	76.1

Table 4: Short-context evaluation on standard benchmarks. Full denotes the original backbone without memory adaptation. Activation Beacon is included as the homogeneous carrier-memory baseline.

C Full Compression-Ratio Scaling Results

Figure 7 extends the main compression-ratio analysis to all five LongBench categories. Strata Memory consistently maintains stronger performance as compression becomes more aggressive. The separation is largest on Multi-Doc and Code, where compressed history must support both cross-segment aggregation and precise evidence recovery. The gains on Summarization and Few-Shot are smaller, but follow the same trend.

D Per-Dataset LongBench Results

Table 5 provides a per-dataset breakdown of the Qwen-2-7B results in Table 1. Strata Memory improves over Activation Beacon on all fourteen LongBench datasets, indicating that the category-level gains are not driven by a small number of outlier tasks. The largest improvements appear on Multi-Doc QA and Code, including HotpotQA, 2WikiMQA, MuSiQue, LCC, and RepoBench-P. This pattern is consistent with the main analysis: role-structured compressed memory is most beneficial when a task requires both cross-context aggregation and local evidence recovery. The gains on Summarization and Few-Shot are smaller but remain positive across all listed datasets.

E Ablation Variant Definitions

We define the ablation variants used in Table 2 as follows. All variants keep the same total carrier budget, compression protocol, training recipe, decoding configuration, and evaluation protocol.

No Role Structure. All carrier states share one untyped carrier interface and are stored in a single homogeneous memory stream. The memory is read with a unified access policy.

Typed Carriers Only. Carrier positions keep Anchor/Detail type assignments and type-aware carrier projections. However, their key-value states are merged into a single memory stream and read with a unified access policy. This variant tests whether typed writing alone is sufficient.

Bank-Specific Access Only. Carrier positions are split into two banks by the same deterministic

Method	Single-Doc QA			Multi-Doc QA			Summarization			Few-Shot			Code	
	NQA	Qasper	MFQA	HotpotQA	2Wiki	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RepoBench-P
Full-FT	27.95	40.44	54.61	52.84	44.15	24.83	31.29	21.17	27.92	72.56	89.05	43.87	67.85	64.36
Act. Beacon	27.01	40.98	53.51	52.58	43.76	24.59	31.25	21.49	27.63	72.34	88.78	44.12	67.97	64.80
Strata Memory	**28.62**	**42.27**	**55.66**	**55.69**	**46.43**	**27.39**	**31.88**	**21.66**	**28.23**	**73.00**	**89.81**	**44.59**	**70.45**	**66.63**

Table 5: Per-dataset LongBench results on Qwen-2-7B. NQA, MFQA, 2Wiki, and RepoBench-P denote NarrativeQA, MultiFieldQA-en, 2WikiMQA, and RepoBench-P, respectively. All scores are computed with the same evaluation protocol as Table 1. Bold indicates the best result for each dataset.

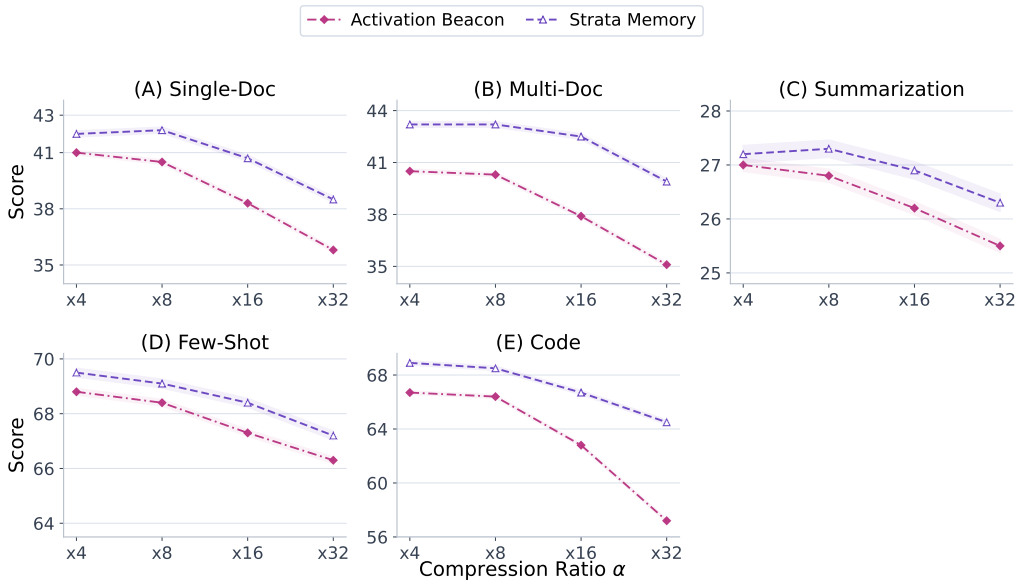

Figure 7: Full compression-ratio scaling results over five LongBench categories. Scores are reported under compression ratios $\alpha \in \{4, 8, 16, 32\}$. Strata Memory degrades more slowly than Activation Beacon, with the largest separation on Multi-Doc and Code.

placement schedule, but they share the same carrier type and carrier projection parameters. Only the read policies differ across the two banks. This variant tests whether asymmetric reading alone is sufficient.

Dual Banks + Uniform Access. Typed carriers are written into Anchor and Detail banks, but both banks are exposed through the same unified read policy. This variant tests whether separating storage is sufficient without bank-consistent access.

Swapped Access Policies. Typed carriers are written into their original banks, but the Anchor bank is read with the Detail policy and the Detail bank is read with the Anchor policy. This variant tests whether the write-time role must be aligned with the read-time policy.

F Single-Role Memory Ablations

Table 6 tests whether either proposed memory role can replace the full role-structured interface. All single-role variants are weaker than Strata Memory across all benchmark categories. Anchor-Only Memory is most affected on Code, where the score drops from 68.5 to 64.9. This suggests that sparse long-range coverage alone is insufficient for tasks that depend on local definitions, identifiers, and short evidence spans.

Detail-Only Memory is less harmful than Anchor-Only Memory on Code, improving from 64.9 to 66.8. However, it remains weaker than the full model, especially on Multi-Doc, where it obtains 39.4 compared with 43.2 for Strata Memory. This pattern is consistent with the intended local-evidence role of Detail Memory, but it does not by itself establish which information is encoded in the bank.

The no-role variant is intentionally different from the single-role variants. It tests a same-budget homogeneous memory interface, whereas Anchor-Only and Detail-Only test whether one proposed role can substitute for the other. The results show that neither role alone is sufficient. Strata Mem-

Method	Single-Doc	Multi-Doc	Summ.	Few-Shot	Code
No Role Structure (same budget)	40.4	40.2	26.7	68.3	66.3
Anchor-Only Memory	39.7	39.8	26.6	67.8	64.9
Detail-Only Memory	40.3	39.4	26.2	67.6	66.8
Strata Memory (Ours)	**42.2**	**43.2**	**27.3**	**69.1**	**68.5**

Table 6: Single-role memory ablations on Qwen-2-7B. All variants keep the same total carrier budget. Anchor-Only and Detail-Only route all compressed carrier states to one memory interface.

ρ	**Single-Doc**	**Multi-Doc**	**Summ.**	**Few-Shot**	**Code**
0.2	40.4	41.1	26.1	68.3	67.6
0.4	41.9	42.7	26.9	69.0	68.5
0.5	42.2	43.2	27.3	69.1	68.4
0.6	42.3	43.2	27.2	69.1	67.8
0.8	40.5	41.9	27.0	68.5	66.9

Table 7: Sensitivity to the anchor-budget ratio ρ on Qwen-2-7B. The total carrier budget is fixed.

ory benefits from combining Anchor Memory and Detail Memory under a fixed carrier-write budget, rather than collapsing compressed states into a single homogeneous stream or a single role-specific interface.

G Anchor Ratio Sensitivity

We study the sensitivity of Strata Memory to the anchor-budget ratio ρ, which controls how the fixed carrier-write budget is divided between Anchor Memory and Detail Memory. We sweep $\rho \in \{0.2, 0.4, 0.5, 0.6, 0.8\}$ on Qwen-2-7B while keeping the total number of carriers unchanged.

Table 7 shows that moderate allocations perform best. Both anchor-light and anchor-heavy settings reduce performance, suggesting that long-context prediction benefits from preserving both global structure and local evidence. The range $\rho = 0.4$–0.6 remains consistently strong, indicating that the method is not tied to a single exact split.

H Memory Budget Definitions

Persistent memory size. For a constant compression factor α, the number of persistent compressed carrier states is approximately

$$N_{\text{mem}} = \sum_{i=1}^{T} k_i \approx \frac{n}{\alpha}. \tag{10}$$

Thus, the persistent historical memory scales with the compressed carrier length rather than the raw sequence length. All main experiments use only the compressed Anchor and Detail banks as persistent historical memory.

Anchor and Detail allocation. For a chunk with $k_i > 1$ carriers, the number of Anchor carriers is defined by Eq. 3, and the remaining $k_i^d = k_i - k_i^a$ carriers are assigned to Detail Memory. Anchor carriers are placed by a deterministic middle-of-bin rule. For $r = 0, \ldots, k_i^a - 1$, define

$$p_i(r) = 1 + \left\lfloor \frac{(r + \frac{1}{2})k_i}{k_i^a} \right\rfloor. \tag{11}$$

The Anchor set is

$$\mathcal{A}_i = \{c_{i,p_i(r)} : r = 0, \ldots, k_i^a - 1\}. \tag{12}$$

The Detail set is

$$\mathcal{D}_i = \{c_{i,1}, \ldots, c_{i,k_i}\} \setminus \mathcal{A}_i. \tag{13}$$

When $k_i = 1$, we set $\mathcal{A}_i = \emptyset$ and $\mathcal{D}_i = \{c_{i,1}\}$.

Readable bank sizes. For each layer ℓ, the Anchor read rule in Eq. 6 gives

$$B_i^a = |S_i^{a,\ell}| = \min(|M_{i-1}^{a,\ell}|, T_a). \tag{14}$$

The Detail read rule in Eq. 7 gives

$$B_i^d = |S_i^{d,\ell}| = \begin{cases} |M_{i-1}^{d,\ell}|, & T_d = \infty, \\ \min(|M_{i-1}^{d,\ell}|, T_d), & \text{otherwise.} \end{cases} \tag{15}$$

The main text omits the layer index in B_i^a and B_i^d because the same read budgets are applied at each layer. For each ordinary query, the memory-side read length is

$$L_i^{\text{read}} = B_i^a + B_i^d. \tag{16}$$

Unless otherwise specified, we use $\rho = 0.5$, $T_a = 256$, and $T_d = \infty$. Sensitivity to ρ and to the read budgets is reported in Appendices G and K, respectively.

T_a	T_d	Single-Doc	Multi-Doc	Summ.	Few-Shot	Code
64	∞	41.3	42.0	27.0	68.8	67.6
128	∞	41.8	42.7	27.2	69.0	68.1
256	∞	42.2	43.2	27.3	69.1	68.5
512	∞	42.3	43.1	27.2	69.0	68.4
256	512	41.9	42.6	27.1	68.9	68.0
256	256	41.4	41.8	26.9	68.7	67.1

Table 8: Sensitivity to Anchor and Detail read budgets on Qwen-2-7B. All rows use the same trained model, total carrier budget, compression ratio, role allocation, and evaluation protocol. Only inference-time read budgets are changed. The default setting used in the main experiments is $T_a = 256$ and $T_d = \infty$.

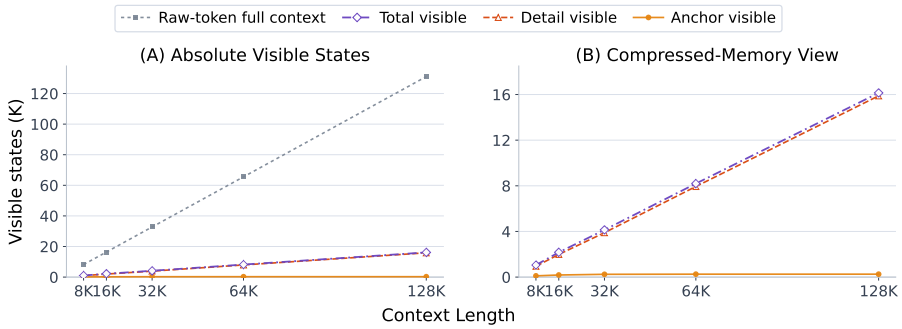

Figure 8: Visible-state growth under the default read policy. We use $\alpha = 4$, $\rho = 0.5$, $T_a = 256$, and $T_d = \infty$. Anchor reads are capped by T_a, while Detail reads remain visible over the compressed carrier history. As a result, the total visible memory length grows with context length, but at the scale of compressed carrier states rather than raw-token full context.

I Visible-State Growth Analysis

We provide a visible-state accounting analysis for the default read policy. This analysis separates the number of compressed carrier states retained in memory from the number of states visible to a query. Let n denote the accumulated historical token length before the current query. For compression ratio α, anchor ratio ρ, Anchor read budget T_a, and Detail read budget T_d, the approximate visible memory length is

$$L^{\text{read}}(n) = \min\left(\rho\frac{n}{\alpha}, T_a\right) + B^d(n), \qquad (17)$$

where

$$B^d(n) = \begin{cases} (1-\rho)\frac{n}{\alpha}, & T_d = \infty, \\ \min\left((1-\rho)\frac{n}{\alpha}, T_d\right), & \text{otherwise.} \end{cases} \qquad (18)$$

Thus, the carrier budget and the visible read length are related but distinct: The former controls how many compressed states are written and retained, while the latter is determined by the bank-specific read policies.

Figure 8 shows this accounting for context lengths up to 128K. The Anchor curve saturates at $T_a = 256$, whereas the Detail curve grows linearly with the compressed Detail history. At 128K context length under $4\times$ compression and $\rho = 0.5$, the default policy exposes roughly 16K Detail states plus the capped Anchor states, compared with 128K raw-token states under full-context attention. This makes the cost behavior explicit: the default read length is not constant, but it grows over compressed carrier memory rather than raw-token history. With finite T_d, Eq. 18 bounds the memory-side read length by $T_a + T_d$.

J Matched Visible-State Budget Comparison

The default read policy separates the written carrier budget from the number of memory states visible to each query. To verify that the improvement does not come from exposing a larger read interface, we compare Strata Memory with the homogeneous no-role variant under matched visible-state budgets. For each budget, both methods expose the same number of visible compressed states per query. The total carrier budget, compression ratio, training recipe, and evaluation protocol are kept unchanged.

Figure 9 shows that Strata Memory consistently

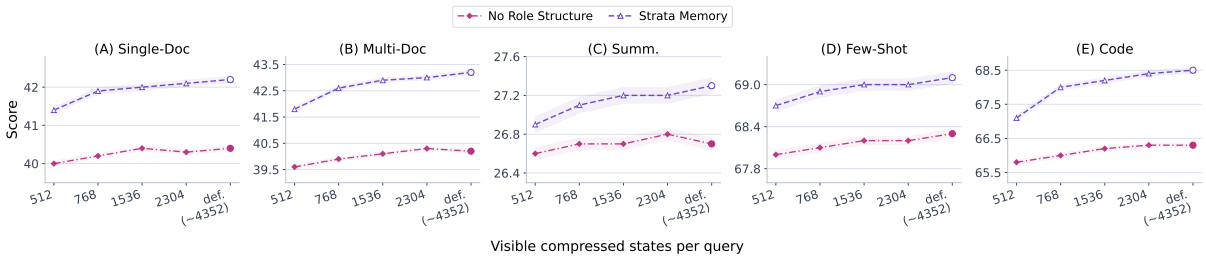

Figure 9: Matched visible-state budget comparison on Qwen-2-7B. The x-axis gives the number of visible compressed states per query, and "default" denotes the visible read length induced by the main setting in 32K LongBench evaluation. At each budget, Strata Memory and the homogeneous no-role variant use the same visible memory length. Strata Memory remains consistently stronger across all tested budgets, with the largest margins on Multi-Doc and Code.

outperforms the homogeneous no-role variant at every matched read budget. The advantage is already present under the tightest budget of 512 visible states per query and remains stable as the visible memory length increases. This rules out the possibility that the gain is simply caused by exposing more memory states at inference time.

The improvement is most pronounced on Multi-Doc and Code, where the average gain across the tested budgets is about 2.7 and 1.9 points, respectively. These are also the categories that most strongly require both cross-context aggregation and local evidence recovery. In contrast, the gain on Summarization is smaller but still consistently positive.

Increasing the visible-state budget benefits both methods, but the homogeneous variant saturates earlier. Strata Memory continues to improve from 512 to the default setting, especially on Multi-Doc and Code. This pattern suggests that the role-consistent write–read interface does not merely increase access volume; it makes the visible compressed states more useful under the same read budget.

K Access-Budget Sensitivity

We study the sensitivity of Strata Memory to inference-time read budgets. The anchor ratio ρ controls how the fixed carrier-write budget is divided during writing, whereas T_a and T_d control how many Anchor and Detail states are visible during reading. We vary T_a and T_d on Qwen-2-7B while keeping the trained model, total carrier budget, compression ratio, role allocation, and evaluation protocol fixed.

Table 8 evaluates whether Strata Memory depends on a narrow read-budget configuration. Increasing T_a from 64 to 256 improves all categories, indicating that too small an Anchor read budget limits long-range coverage. Further increasing T_a to 512 gives only marginal, category-specific changes: Single-Doc increases slightly from 42.2 to 42.3, while Multi-Doc, Summ., Few-Shot, and Code do not improve over the default setting. This suggests that the default $T_a = 256$ provides sufficient sparse coverage without relying on an extreme Anchor read budget.

Capping Detail Memory produces a clearer degradation. Reducing T_d from ∞ to 512 lowers all categories slightly, and the stronger cap $T_d = 256$ leads to larger drops. The largest decreases occur on Multi-Doc and Code, where scores fall from 43.2 to 41.8 and from 68.5 to 67.1, respectively. This sensitivity pattern is consistent with the intended local-evidence-oriented role of Detail Memory, but it should be interpreted as budget-sensitivity evidence rather than causal proof of the exact information stored in the bank.

L Reproducibility Details

Cue-range diagnostic. Figure 2 is computed from measured cue instances in our analysis set. The displayed tokens and phrases are representative examples selected for readability from the measured cues, not manually designed schematic labels. We group cues into structural cues, such as topic shifts, entity chains, speaker turns, and file dependencies, and local-evidence cues, such as dates, line references, identifiers, and article references. For each cue instance, we measure its future-use profile over chunk distance and aggregate profiles within each cue group. The shaded regions indicate variation across measured cue instances. This diagnostic is used only for analysis and motivation; cue

labels are not used for training, role assignment, or retrieval supervision.

Models and trainable parameters. We evaluate Strata Memory in a frozen-backbone setting. Backbone parameters are kept fixed, and optimization is restricted to memory-related parameters, including carrier embeddings, carrier-specific projections, and type-aware scaling parameters. The main experiments use Llama-2-7B-Chat and Qwen-2-7B-Instruct, with additional evaluations on Qwen-2.5-7B and Qwen-2.5-14B. The chunk size is $w = 1024$ for Llama-2 and $w = 2048$ for Qwen-family models.

Compression and decoding settings. For the main LongBench comparisons, compressed-context methods follow the same compression protocol as the corresponding activation-compression baseline. Llama-2 uses adaptive compression, with $2\times$ compression for 4K–8K contexts, $4\times$ compression for 8K–16K contexts, and $8\times$ compression for 16K–32K contexts. Qwen-family models, including Qwen-2-7B, Qwen-2.5-7B, and Qwen-2.5-14B, use uniform $4\times$ compression. Efficiency-oriented evaluations use $8\times$ compression unless otherwise specified. All controlled comparisons use the same original input length and decoding configuration within each backbone setting.

Training recipe. Training follows a two-stage memory adaptation recipe. The first stage trains the memory modules on long plain-text sequences sampled from RedPajama. The second stage adapts the memory modules on long-context instruction and summarization data, including LongAlpaca, BookSum, and synthetic long-context supervision. All training samples are shorter than 20K tokens. We use bf16 precision, a batch size of 8, and linear learning-rate decay without warmup. The learning rate is 5×10^{-5} in the first stage and 1×10^{-5} in the second stage.

Same-budget variants. For controlled ablations and comparisons with Activation Beacon, Strata Memory uses the same total carrier budget and training recipe. Variants differ only in typed carrier allocation, memory organization, and bank-specific access, rather than in the number of compressed states. Unless otherwise specified, we use $\rho = 0.5$, $T_a = 256$, and $T_d = \infty$. Sensitivity to ρ and the read budgets is reported in Appendices G and K.

Example code. To further support reproducibility, we provide anonymized example code as a ZIP archive in the Software field. The example code is intended as a minimal reference implementation of the Strata Memory interface.

M Artifact Use and Licensing

We use only publicly released third-party artifacts, including the models, datasets, benchmarks, and baselines listed in Section 4.1, for research-purpose model adaptation and evaluation. We follow the licenses and terms stated by the original artifact providers and do not redistribute third-party data, model weights, or baseline code. Any released artifacts from this work are intended only for research on long-context compression and are not intended for sensitive or user-facing deployment.