Bridging the Gap: Position-Independent Cache Reuse for Hybrid SSM-Attention Architectures

Weikang Wang[*], **Xin Zhou**[*], **Haiyang Liu, Lei Wang, Jun Liu, Weifeng Zhang**
Intelligent Computing Infrastructure Lab, Lenovo Research, China
{wangwk12,zhouxin16,liuhy50,wanglei106,liujun8,weifengz}@lenovo.com

Abstract

Hybrid large language models (LLMs) integrating state space models (SSMs) and full attention face a critical caching bottleneck due to mismatched caching granularities. While full attention allows flexible, token-level cache reuse, SSMs are always restricted to coarse-grained block- or request-level reuse. Consequently, an SSM layer's cache miss can easily invalidate successful fine-grained hits in adjacent attention layers, disrupting the entire reuse pipeline. To resolve this, chunk-level position-independent cache (PIC) reuse is highly desirable to align the caching granularity across all layers. However, enabling PIC reuse for history-dependent SSMs remains fundamentally challenging. To bridge this gap, we propose *HyPIC*, the first unified framework achieving PIC reuse for hybrid architectures. We introduce residual state propagation and online decay correction, which efficiently adapt offline SSM chunk caches to new contexts by explicitly recomputing minimal boundary tokens. Furthermore, we design a fast parallel recomputation pipeline exploiting the linear superposition property of SSMs to eliminate sequential bottlenecks. Experiments on diverse hybrid models and datasets demonstrate that HyPIC achieves a $2\times$ speedup in time-to-first-token (TTFT) while maintaining highly competitive accuracy. Code is available at: https://github.com/Lenovo-ICI/HyPIC.

1 Introduction

The rapid evolution of large language models (LLMs) [1–4] has driven a growing demand for long-context modeling, particularly in emerging applications such as agentic systems [5–7] and retrieval-augmented generation (RAG) [8–10]. However, standard transformer architectures based on full attention suffer from quadratic computational complexity, causing severe resource bottlenecks in long-context scenarios. To address this, state space models (SSMs) [11] have been proposed to compress sequence history into fixed-size hidden states, achieving linear time complexity. While highly efficient, pure SSMs tend to suffer from inferior performance due to the compression of historical context. Recently, hybrid architectures combining both SSMs and full attention (e.g., Nemotron-H [12], Qwen3.5 [13], and Kimi-Linear [14]) have emerged as a promising paradigm, offering a favorable balance between computational efficiency and long-context performance.

A promising direction to mitigate the computational bottleneck in long-context inference is cache reuse, which aims to eliminate redundant computation by leveraging previously computed intermediate representations. In transformer-based models with full attention, key-value (KV) cache reuse has been extensively studied and widely adopted. Techniques such as prefix reuse [15, 16] and position-independent cache (PIC) reuse [17–19] enable flexible composition of historical contexts, significantly reducing time-to-first-token (TTFT) during inference by avoiding redundant processing of shared text segments. For pure SSMs, prefix caching is similarly straightforward, allowing systems to achieve substantial speedups by simply reusing the hidden states at the end of a shared prefix,

[*]Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

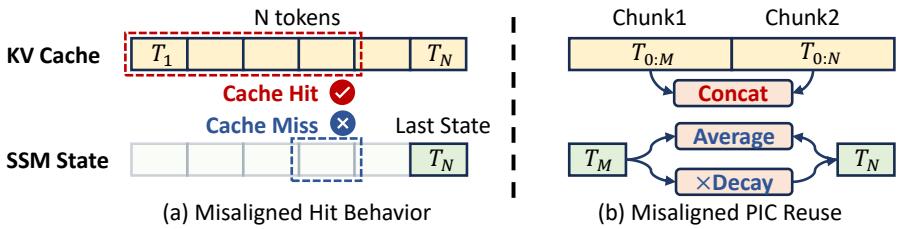


Figure 1: The misalignment bottleneck in hybrid model cache reuse. T_i denotes the cached state for the i-th token, and $T_{i:j}$ represents the chunk-level cache for the sequence from token i to j.

thereby avoiding the recomputation of prior tokens. However, despite the success of these individual paradigms, extending cache reuse to hybrid models presents a major challenge. Their heterogeneous architectures introduce severe structural conflicts that fundamentally hinder efficient cache sharing.

The primary challenge of cache reuse in hybrid models lies in the granularity mismatch between SSM and full-attention layers. While the KV cache enables token-level reuse, the substantial memory overhead of SSM hidden states restricts their reuse to coarse-grained block- or request-level. As illustrated in Figure 1(a), this misalignment of hit granularity creates a critical bottleneck: even when a subset of tokens successfully hit the KV cache in attention layers, the inability to meet the coarse-grained matching condition in SSM layers results in a cache miss. This motivates us to align the caching granularity by adopting chunk-level position-independent caching (PIC). However, PIC mechanisms for SSMs remain largely unexplored. As shown in Figure 1(b), while independent KV caches from different chunks can be seamlessly concatenated along the sequence dimension, the assembly of SSM hidden states is highly non-trivial. Existing approaches such as Soup [20] and PICASO [21] directly average the SSM states for fusion, with Soup further employing an "A-Decay" method that propagates states across chunks using summations weighted by decay matrices. Such naive fusion strategies struggle to accurately capture the temporal dependencies across chunks, leading to suboptimal performance.

To bridge this gap, we propose HyPIC, the first unified framework to enable PIC reuse in hybrid architectures. Specifically, we introduce residual state propagation, which recomputes a minimal set of boundary tokens to restore inter-state information flow and minimize propagation errors. To further preserve accuracy, we apply an online decay correction mechanism that mitigates decay variance and prevents cross-chunk error accumulation. Finally, to maximize efficiency, we design a fast parallel pipeline that exploits the linear superposition property of SSMs, eliminating sequential bottlenecks.

Our main contributions are summarized as follows:

- We propose HyPIC, a unified framework to resolve the caching granularity mismatch in hybrid architectures, successfully enabling PIC reuse for models combining full-attention and SSM layers.
- We introduce a residual state propagation strategy coupled with an online decay correction mechanism. By explicitly recomputing a minimal set of boundary tokens, this approach effectively restores temporal information flow across chunks and suppresses decay variance, thereby preventing cross-chunk error accumulation.
- We develop a fast parallel recomputation pipeline that exploits the linear superposition property of modern SSMs. By decoupling state injection from recurrent scans, this pipeline completely eliminates sequential bottlenecks and enables highly efficient state assembly.
- Extensive experiments across diverse hybrid models and datasets demonstrate that HyPIC achieves a 2× speedup in TTFT while maintaining highly competitive accuracy.

2 Related Work

2.1 LLM Architectures

Standard LLMs based on the full attention mechanism [22] have achieved remarkable success in natural language processing. However, since full attention computes pairwise token affinities, it incurs quadratic computational complexity of $O(L^2)$ and a linearly growing KV cache memory footprint.

This fundamental limitation severely restricts their scalability in ultra-long context scenarios, such as agentic workflows. To overcome the quadratic bottleneck, researchers have shifted their focus towards linear-time sequence models [23], particularly SSMs (e.g., Mamba [11, 24] and GatedDeltaNet [25]), which achieve an $O(1)$ inference memory footprint per token by compressing historical context into fixed-size hidden states. Despite their efficiency, pure SSMs suffer from inherent information loss because their fixed-size states inevitably fail to retain fine-grained historical context, leading to suboptimal generation quality. To mitigate this, recent research has increasingly turned to hybrid architectures [12, 14, 26, 27]. By combining predominantly SSM layers for computational efficiency with a smaller number of full-attention layers for fine-grained information retrieval, these models achieve an optimal performance-efficiency trade-off. Nevertheless, their architectural heterogeneity introduces fundamental structural conflicts when deploying advanced inference acceleration techniques, particularly for cache reuse.

2.2 Cache Reuse in LLMs

Prefix Cache. To reduce prefilling latency, prefix caching has been widely adopted in LLMs. For full-attention models, KV cache reuse and management [28, 15, 16] has become a well-established strategy, which minimizes redundant computation by maintaining the KV cache of shared prefixes across multiple requests. Similarly, for pure SSMs, prefix caching can be implemented by saving and restoring the hidden states at the exact boundary of a shared prefix [15, 16]. Accordingly, prefix caching in hybrid models naturally integrates these two mechanisms [29], where the overall hit length is strictly determined by the longest common prefix across both full-attention and SSM layers.

PIC for full-attention models. To overcome the strict prefix matching limitation, recent studies have introduced PIC. By decoupling KV caches from their absolute prompt positions, PIC enables flexible context composition. These mechanisms for full-attention models have been thoroughly explored and can be broadly classified into two categories: (1) fine-tuning methods [30, 31] that adapt LLMs to process isolated KV caches, and (2) recomputation methods [18, 19, 17] that selectively recompute critical tokens to reconstruct inter-chunk correlations. Despite their differences, both strategies rely on directly concatenating independent KV caches and explicitly restoring positional consistency. This token-level modularity makes PIC exceptionally effective in full-attention settings.

PIC for SSMs. Extending PIC to SSMs is highly challenging. Unlike token-level KV representations in full-attention models, the hidden states in SSMs inherently encode strict sequence-level historical dependencies. Consequently, existing approaches mostly rely on simple heuristic approximations. For instance, Soup [20] attempts to propagate independent states across chunks by employing simple state averaging or an "A-decay" mechanism driven by the decay matrix. Similarly, PICASO [21] addresses multi-chunk context composition by directly applying mean-pooling over isolated SSM hidden states to form a fused representation. However, such naive fusion strategies severely disrupt the continuous temporal dynamics of SSMs, leading to suboptimal generation performance.

3 Preliminaries

Hybrid SSM-attention architectures combine full-attention layers with SSM layers to balance modeling capability and computational efficiency. While full-attention layers maintain token-level KV caches, SSM layers compress historical information into recurrent hidden states. Specifically, modern SSMs [11, 25] parameterize sequence mappings through a data-dependent linear time-varying recurrence. At time step t, given the current input $\boldsymbol{x}_t$ and the previous hidden state $\mathbf{S}_{t-1}$, the hidden state evolves as:

$$\mathbf{S}_t = \bar{\mathbf{A}}_t \mathbf{S}_{t-1} + \bar{\mathbf{B}}_t \boldsymbol{x}_t, \tag{1}$$

where $\bar{\mathbf{A}}_t$ and $\bar{\mathbf{B}}_t$ are the discretized state transition and input projection matrices, respectively. We refer to $\bar{\mathbf{A}}_t$ as the stepwise decay. For $t_1 \leq t_2$, we define $\bar{\mathbf{A}}_{t_1:t_2} = (\bar{\mathbf{A}}_{t_1}, \ldots, \bar{\mathbf{A}}_{t_2})$ as the sequence of stepwise decays and $\mathbf{D}_{t_1:t_2} = \prod_{j=t_1+1}^{t_2} \bar{\mathbf{A}}_j$ as the cumulative decay from state t_1 to state t_2.

This recurrent dependence complicates PIC reuse: the SSM state of a chunk depends on the context preceding it. In structured workloads such as RAG and agentic workflows, prompts typically contain independently reusable chunks with predefined boundaries (e.g., retrieved documents or modular prompt components). Consider a reusable chunk C of length L and its preceding context P. For

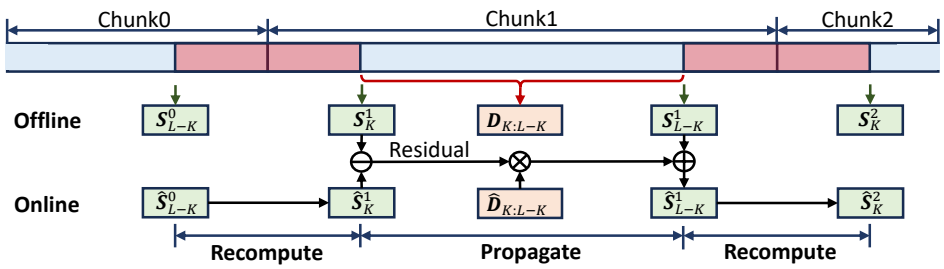


Figure 2: Overview of the state propagation pipeline in HyPIC.

$1 \leq t \leq L$, we denote the state obtained by processing C independently as $\mathbf{S}_t = \mathrm{SSM}(C)_t$, and the corresponding context-aware state as $\mathbf{S}_t^\star = \mathrm{SSM}(P \oplus C)_{|P|+t}$, where $\oplus$ denotes sequence concatenation. These states generally differ because $\mathbf{S}_t^\star$ incorporates the preceding context. SSM-layer PIC reuse therefore aims to recover the context-aware states needed for inference from independently cached chunk information without fully recomputing the chunk. We denote the approximately recovered context-aware state by $\hat{\mathbf{S}}_t$.

4 Method

Figure 2 illustrates the pipeline of HyPIC, which consists of two phases: offline generation and online recomputation. In the offline phase, the SSM states and KV caches of each independent chunk (e.g., knowledge chunks in RAG) are computed and stored. During online inference, these cached chunks are assembled into the current context, and a unified recomputation window of size $2K$ is applied to each chunk (i.e., the first and last K tokens). The two types of layers are then processed as follows:

- **Full-attention layers.** We directly concatenate the offline KV caches of all chunks and explicitly recompute the aforementioned $2K$ boundary tokens to update the caches.
- **SSM layers.** For each chunk of length L, we store two boundary states $\mathbf{S}_K$ and $\mathbf{S}_{L-K}$, the stepwise decays $\bar{\mathbf{A}}_{1:K}$, and the cumulative decay $\mathbf{D}_{K:L-K}$. During online recomputation, after explicitly recomputing the same boundary tokens, the states of the remaining intermediate tokens are efficiently updated using our residual state propagation strategy.

Given that the processing of full-attention layers is relatively standard [18, 17, 19], the following subsections focus primarily on our design for SSM layers.

4.1 Residual State Propagation

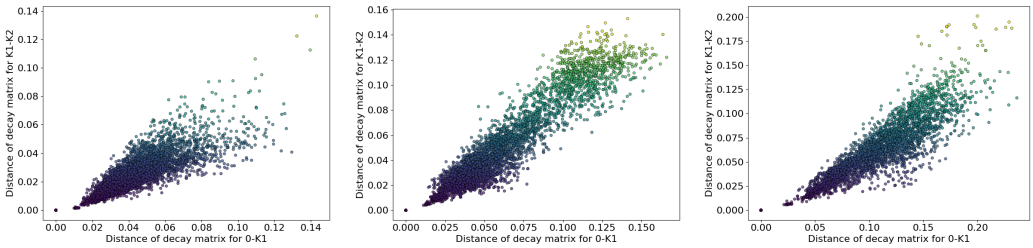
To efficiently reconstruct the context-aware SSM states from independently cached chunks, we exploit the linear recurrence of SSMs. Specifically, for a chunk of length L, we can unroll the state recurrence from an initial step t_0 to the final step $t = t_0 + L$, yielding:

$$\mathbf{S}_t = \left(\prod_{i=t_0+1}^{t} \bar{\mathbf{A}}_i \right) \mathbf{S}_{t_0} + \sum_{j=t_0+1}^{t} \left(\prod_{i=j+1}^{t} \bar{\mathbf{A}}_i \right) \bar{\mathbf{B}}_j \boldsymbol{x}_j. \tag{2}$$

Using the cumulative decay $\mathbf{D}_{t_0:t}$ defined above, let $\mathbf{I}_{t_0:t}$ denote the cumulative input term. We can rewrite the state propagation compactly as:

$$\mathbf{S}_t = \mathbf{D}_{t_0:t} \mathbf{S}_{t_0} + \mathbf{I}_{t_0:t}. \tag{3}$$

In PIC reuse scenarios, processing a text chunk independently (e.g., during offline generation) yields a reference final state $\mathbf{S}_t$, derived from an isolated initial state $\mathbf{S}_{t_0}$. However, when this chunk is assembled with preceding context during online inference, the target context-aware state is $\mathbf{S}_t^\star$. We aim to recover an approximation $\hat{\mathbf{S}}_t$ from a context-aware initial state $\hat{\mathbf{S}}_{t_0}$. Since the cumulative decay matrix $\mathbf{D}_{t_0:t}$ and the input term $\mathbf{I}_{t_0:t}$ are predominantly determined by linear projections of local chunk inputs, we introduce a practical approximation: both terms remain largely invariant to the preceding context (i.e., $\mathbf{I}_{t_0:t} \approx \hat{\mathbf{I}}_{t_0:t}$ and $\mathbf{D}_{t_0:t} \approx \hat{\mathbf{D}}_{t_0:t}$). Consequently, to compute $\hat{\mathbf{S}}_t$ without fully



(a) Nemotron-H-8B-Reasoning-128K (b) Qwen3.5-9B (c) Kimi-Linear-48B-A3B-Instruct

Figure 3: Linear correlation of decay errors between chunk prefixes and their subsequent regions. This linear trend suggests that prefix divergence serves as a partial predictor for future decay errors.

recomputing the entire chunk, we subtract the offline reference equation from the online equation. This cancels the complex input term, yielding a highly efficient residual approximation:

$$\hat{\mathbf{S}}_t = \mathbf{S}_t + \mathbf{D}_{t_0:t} \left(\hat{\mathbf{S}}_{t_0} - \mathbf{S}_{t_0} \right). \tag{4}$$

Building upon this formulation, Figure 2 illustrates its practical application within our pipeline. The propagation spans from the initial boundary at step K to the final boundary at step $L-K$. Consequently, to recover the approximate online context-aware state $\hat{\mathbf{S}}^i_{L-K}$ for the i-th chunk, we directly apply our residual propagation rule (Equation 4). Specifically, we use the offline-generated reference states $\mathbf{S}^i_K$ and $\mathbf{S}^i_{L-K}$, the cumulative decay matrix $\mathbf{D}^i_{K:L-K}$, and the newly recomputed online initial state $\hat{\mathbf{S}}^i_K$.

4.2 Online Decay Correction

While Equation 4 is highly efficient, its accuracy is fundamentally limited by the approximated cumulative decay. Since this decay term determines the amount of historical information retained at the current step, any estimation error can lead to severe performance degradation. Consequently, further correcting the cumulative decay $\mathbf{D}_{K:L-K}$ is essential for accurate state propagation. To address this, we propose a correction strategy that exploits the explicit relationship between the offline and online prefix decays.

Specifically, for each chunk i, we first extract the sequence of offline decay matrices $\bar{\mathbf{A}}^i_{1:L-K}$ and its online counterpart $\hat{\mathbf{A}}^i_{1:L-K}$. We then compute the L2 distance between these two sequences, comparing the explicitly recomputed prefix (tokens $1 \to K$) with the subsequent intermediate region (tokens $K+1 \to L-K$), which can be quantified as $s_{\text{ini}}(i) = \|\bar{\mathbf{A}}^i_{1:K} - \hat{\mathbf{A}}^i_{1:K}\|_2$ and $s_{\text{sub}}(i) = \|\bar{\mathbf{A}}^i_{K+1:L-K} - \hat{\mathbf{A}}^i_{K+1:L-K}\|_2$. We conduct this empirical analysis on HotpotQA [32] using three hybrid models: Nemotron-H-8B-Reasoning-128K [12], Qwen3.5-9B [13], and Kimi-Linear-48B-A3B-Instruct [14]. As illustrated in Figure 3, we observe an approximate linear correlation between the decay errors in the prefix tokens and the errors in the subsequent intermediate tokens. This finding indicates that subsequent decay errors can be partially predicted and compensated for by analyzing prefix discrepancies (see Appendix B.1 for detailed analysis).

Building on this insight, we propose a geometric correction pipeline. Rather than directly extrapolating the raw decay error, which may lead to instability, we decouple the decay into magnitude and direction. This formulation enables independent control over the extrapolation step sizes governing global decay intensity and channel-wise feature contrast. Specifically, given the stepwise offline decay matrices $\bar{\mathbf{A}}^i_{1:K}$ and the recomputed online counterpart $\hat{\mathbf{A}}^i_{1:K}$, we decompose each matrix at step k into a scalar magnitude $(m^i_k, \hat{m}^i_k)$ and a unit directional vector $(\boldsymbol{u}^i_k, \hat{\boldsymbol{u}}^i_k)$. The magnitude discrepancy is simply measured by the scalar ratio $r^i_k = \hat{m}^i_k / m^i_k$. For the directional shift $\Delta\boldsymbol{u}^i_k$, motivated by the exponential discretization of modern SSMs, we perform an elementwise division $\Delta\boldsymbol{u}^i_k = \hat{\boldsymbol{u}}^i_k / \boldsymbol{u}^i_k$. This division mathematically corresponds to an exact linear subtraction in the continuous log-domain, effectively yielding the directional deviation vector. Finally, we average these deviations across the prefix window to obtain the mean magnitude and directional corrections for each chunk, $\bar{r}^i = \mathbb{E}[r^i_k]$ and $\Delta\bar{\boldsymbol{u}}^i = \mathbb{E}[\Delta\boldsymbol{u}^i_k]$ (see Appendix B.2 for more details).

To apply this correction, we similarly decompose the unadjusted offline cumulative decay $\mathbf{D}^i_{K:L-K}$ into its magnitude m_{target} and direction $\boldsymbol{u}_{\text{target}}$. The final corrected online decay is then formulated as:

$$\hat{\mathbf{D}}^i_{K:L-K} = m_{\text{target}}(\bar{r}^i)^{\tau_m} \operatorname{Norm}\left(\boldsymbol{u}_{\text{target}} \odot (\Delta\bar{\boldsymbol{u}}^i)^{\tau_u}\right), \tag{5}$$

where $\operatorname{Norm}(\cdot)$ denotes L2 normalization, $\odot$ represents the Hadamard product, and τ_u and τ_m are scaling exponents controlling the temporal strength of the correction (see Appendix C.1 for detailed tuning). Integrating this calibrated decay into the propagation formula yields the final state:

$$\hat{\mathbf{S}}^i_{L-K} = \mathbf{S}^i_{L-K} + \hat{\mathbf{D}}^i_{K:L-K}\left(\hat{\mathbf{S}}^i_K - \mathbf{S}^i_K\right). \tag{6}$$

4.3 Fast Parallel Recomputation

While the proposed method is effective in preserving accuracy, its reliance on sequentially recomputing chunk-boundary tokens and performing step-by-step propagation introduces severe latency. Specifically, executing M sequential SSM scans for M chunks incurs substantial Python loop overhead and GPU kernel launch delays, creating a significant bottleneck for TTFT. To overcome this, we exploit the core *linear superposition property* of SSMs [11, 24]. For an SSM step, the hidden state $\mathbf{S}$ can be mathematically decomposed into a data-driven component and a state-driven component:

$$\mathbf{S}_{\text{target}} = \mathbf{U}_{\text{zero-state}} + \mathbf{D}_{\text{cum}} \cdot \mathbf{S}_{\text{init}}, \tag{7}$$

where $\mathbf{U}_{\text{zero-state}}$ represents the input-driven zero-state response, and $\mathbf{D}_{\text{cum}} \cdot \mathbf{S}_{\text{init}}$ denotes the propagated historical state. For M reusable chunks, we assemble their cross-boundary token representations into a batched tensor $\mathbf{H}_{\text{bndy}} \in \mathbb{R}^{M\times 2K\times d}$, where d is the input hidden dimension. This enables the boundary windows to be processed together in batched SSM scans. Leveraging this decoupling, we design a fast parallel recomputation pipeline to reduce $O(M)$ sequential kernel launches to $O(1)$ parallel operations (see Appendix C.2 for more details):

- **Initial zero-state scan.** We construct a $2K$-length cross-boundary window for each chunk i by concatenating the suffix of chunk $i-1$ (tokens $L-K \to L$) and the prefix of chunk i (tokens $1 \to K$). Initializing these batched sequences with $\mathbf{0}$, we launch a single parallel SSM scan (e.g., `mamba_chunk_scan`). This dependency-free operation efficiently extracts the zero-state intermediate output $\mathbf{U}^i_K$ and the cumulative decay $\hat{\mathbf{D}}^i_{-K:K}$ for all chunks simultaneously.
- **State propagation.** This step explicitly recovers the true boundary anchor states for all chunks. By replacing expensive sequential kernel executions with a lightweight tensor loop, we iteratively propagate the preceding anchor state $\hat{\mathbf{S}}^{i-1}_{L-K}$. We first superimpose the data-driven response over the $2K$ window to obtain the true prefix state: $\hat{\mathbf{S}}^i_K = \mathbf{U}^i_K + \hat{\mathbf{D}}^i_{-K:K} \cdot \hat{\mathbf{S}}^{i-1}_{L-K}$. Subsequently, we apply our decay correction (Equation 6) to propagate the state from $\hat{\mathbf{S}}^i_K$ to the next anchor state $\hat{\mathbf{S}}^i_{L-K}$. This purely tensor-based loop chains the boundary states across all M chunks at negligible latency.
- **Final parallel scan.** Having recovered the true initial anchor state $\hat{\mathbf{S}}^{i-1}_{L-K}$ for every chunk, we inject it as the starting state for its corresponding $2K$ cross-boundary window. We then launch a final parallel SSM scan over these boundary windows. Conditioned on the corrected initial states, this scan generates the exact sequence outputs for the boundary tokens, which can then be seamlessly passed to the next layer.

5 Experiments

5.1 Setup

Datasets and metrics. Following [18, 17], we evaluate HyPIC on five diverse datasets from LongBench [33]. These include three QA datasets (MuSiQue [34], HotpotQA [32], and NarrativeQA [35]), one few-shot learning dataset (SAMSum [36]), and one code completion dataset (LCC [37]). Each dataset contains 200 queries. Context construction varies across tasks according to their characteristics. For the QA datasets, we split the text into 512-token chunks using LangChain and retrieve the top-5 chunks as the final context. For the SAMSum dataset, we extract five consecutive 512-token chunks directly from the beginning of the context. Finally, for the LCC dataset, to preserve context

Table 1: Comparison of the structures of different hybrid models. H and D are the number of heads and dimensions in the SSM layers.

Models	SSM		Full Attention	
	Update Rule & Saved Decay	Decay Size	Pos Enc	Operation
Nemotron-H	$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$	$(K+1)H$	NoPE	Concat
Qwen3.5	$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \beta_t (\boldsymbol{v}_t - \alpha_t \mathbf{S}_{t-1} \boldsymbol{k}_t) \boldsymbol{k}_t^T$	$(K+1)H$	MRoPE	Concat+Rotate
Kimi-Linear	$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t) \mathbf{S}_{t-1} + \beta_t (\boldsymbol{v}_t - \mathrm{Diag}(\boldsymbol{\alpha}_t) \mathbf{S}_{t-1} \boldsymbol{k}_t) \boldsymbol{k}_t^T$	$(K+1)HD$	NoPE	Concat

integrity, we partition the entire code context into five chunks of approximately equal length. To further evaluate long-context performance, we also include two tasks from InfiniteBench [38]: Passkey and LongBookQA. Each prompt consists of 50 chunks of 1,024 tokens, yielding over 50K tokens in total. For Passkey, the target information is embedded in one chunk, whereas for LongBookQA, the 50 most relevant chunks are retrieved using vector search.

To evaluate generation quality, we adhere to the LongBench evaluation protocol. Specifically, we use the F1-score for QA datasets to measure the word-level overlap between model outputs and ground-truth answers. For the SAMSum dataset, Rouge-L is used to assess summarization quality based on the longest common subsequence. Furthermore, for the LCC dataset, we employ Edit Similarity to evaluate the character-level alignment between the generated code completions and the reference code. For InfiniteBench, we follow its evaluation protocol, using accuracy for Passkey and Rouge-F1 for LongBookQA. For efficiency, we report the time-to-first-token (TTFT) to measure the latency of producing the first output token.

Implementation details. All experiments are conducted on a server equipped with two NVIDIA H20 GPUs (96 GB VRAM each) and 2 TB DDR5 RAM, with offline caches offloaded to host memory. To conduct a thorough evaluation, we adopt three representative hybrid architectures: Nemotron-H [12] (Mamba2 [24] + Full Attention), Qwen3.5 [13] (Gated Delta Network [25] + Full Attention), and Kimi-Linear [14] (Kimi Gated Attention + Full Attention). We implement HyPIC with Hugging Face Transformers [39] (v4.57.1 for Kimi-Linear and v5.3.0 for the others). As shown in Table 1, for the Mamba2 block in Nemotron-H, the decay term is inherently a scalar for each attention head, which we can directly use for state propagation. However, for Delta Network-based structures (i.e., Qwen3.5 and Kimi-Linear), the strict state update rule involves complex interactions with the previous state $\mathbf{S}_{t-1}$. Storing the complete components required for exact state propagation would incur a prohibitive memory footprint and online recomputation cost. Therefore, to ensure high efficiency, we approximate the transition process by storing and propagating only the primary decay factor (highlighted as α_t and $\mathrm{Diag}(\boldsymbol{\alpha}_t)$ in Table 1). Specifically, the cached decay components comprise two elements: K stepwise decay matrices $\bar{\mathbf{A}}_{1:K}$ for computing the online correction term, and a single cumulative decay matrix $\mathbf{D}_{K:L-K}$ for residual state propagation. This results in $K+1$ stored decay values per chunk. Therefore, for Nemotron-H and Qwen3.5, which employ a scalar decay per attention head, the total storage cost is $(K+1)H$. Conversely, for Kimi-Linear, which applies dimension-specific decays, the storage footprint increases to $(K+1)HD$.

In the full attention layers of these hybrid models, state reuse strategies are determined by their positional encoding schemes (see Table 1). For Nemotron-H and Kimi-Linear, which use NoPE, offline KV caches can be directly concatenated to assemble the context. In contrast, because Qwen3.5 employs MRoPE [40], its cached KV tensors are coupled with absolute positional information. Therefore, after concatenating the caches across different chunks, we perform position re-encoding [30, 31] to properly align the token positions (see Appendix C.2 for more details).

Baselines. We evaluate HyPIC against three baselines:

- **Full Recompute**: The standard inference paradigm, which computes all hidden states from scratch without reusing any historical cache during the prefill stage.
- **Average** [21, 20]: A heuristic cache reuse approach that simply averages the independently computed states across all chunks.
- **A-Decay** [20]: A state propagation baseline that updates the independently reused cache via the transition equation $\hat{\mathbf{S}}_t = \mathbf{S}_t + \mathbf{D}_{t_0:t} \mathbf{S}_{t_0}$.

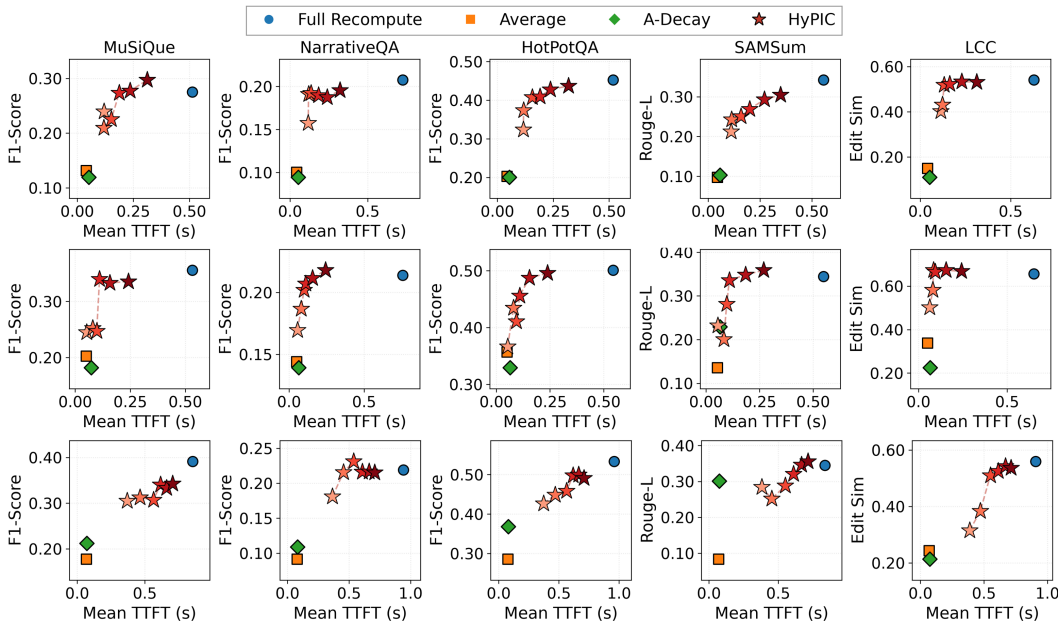


Figure 4: Overall comparison against baselines. Models evaluated (top to bottom): Nemotron-H-8B-Reasoning-128K, Qwen3.5-9B, and Kimi-Linear-48B-A3B-Instruct. Darker shades in "HyPIC" indicate larger recomputation window sizes, $K \in \{4, 8, 16, 32, 64, 98\}$.

For consistency with our approach, the KV caches in the full attention layers are directly concatenated along the sequence dimension for all baselines.

5.2 Main Results

HyPIC and all baselines are evaluated on three representative small-scale hybrid models with fewer than 10B active parameters: Nemotron-H-8B-Reasoning-128K, Qwen3.5-9B, and Kimi-Linear-48B-A3B-Instruct. To validate the robustness of our method across different scales and architectures, we extend our evaluation to a larger-scale hybrid model (Qwen3.5-27B) and a pure Mamba2-based SSM (Mamba-Codestral-7B-v0.1). We also test HyPIC's sensitivity to different recomputation window sizes: $K \in \{4, 8, 16, 32, 64, 98\}$.

Overall Performance. As illustrated in Figure 4, although the Average and A-Decay baselines achieve lower TTFT, they suffer from unacceptable degradation in generation quality. In contrast, Full Recompute yields the highest accuracy but incurs a severe latency penalty. HyPIC (denoted by the connected star markers) strikes an optimal balance between generation quality and efficiency. We observe that generation quality steadily improves as the recomputation window size K increases. Notably, at $K = 64$, HyPIC delivers performance comparable to Full Recompute while maintaining an average $2\times$ speedup in TTFT. This favorable accuracy-efficiency trade-off also extends to longer contexts. As shown in Table 3, HyPIC matches Full Recompute on Passkey and substantially outperforms Average and A-Decay on LongBookQA, while achieving up to a $2.5\times$ TTFT speedup, demonstrating its effectiveness.

Table 2: Comparison of HyPIC with baselines on the LCC dataset using the Mamba-Codestral-7B-v0.1 model.

Methods	Edit Sim	TTFT (s)
Full Recompute	0.6089	0.8569
Average	0.2928	0.0479
A-Decay	0.3132	0.0750
HyPIC	0.6451	0.3048

Additionally, we observe that our acceleration is more pronounced on dense models (Nemotron-H and Qwen3.5) than on the mixture-of-experts (MoE) architecture (Kimi-Linear). This discrepancy arises

Table 3: Results on InfiniteBench with $K = 128$.

		Score				TTFT (s)			
Dataset	Model	Full	Average	A-Decay	HyPIC	Full	Average	A-Decay	HyPIC
Passkey	Qwen3.5	1.0000	0.0000	0.0000	**1.0000**	5.5624	0.2201	0.3525	2.6708
	Nemotron-H	1.0000	0.0000	0.0000	**1.0000**	6.8715	0.2714	0.3505	2.9243
	Kimi-Linear	1.0000	0.0000	0.0000	**1.0000**	6.8950	0.2775	0.3346	4.6052
LongBookQA	Qwen3.5	0.3864	0.1690	0.1665	**0.3215**	5.7505	0.1770	0.3465	2.5811
	Nemotron-H	0.3780	0.0298	0.0339	**0.3078**	7.0514	0.2680	0.4710	2.7953
	Kimi-Linear	0.4059	0.0103	0.1744	**0.3768**	7.1754	0.1630	0.3120	4.9400

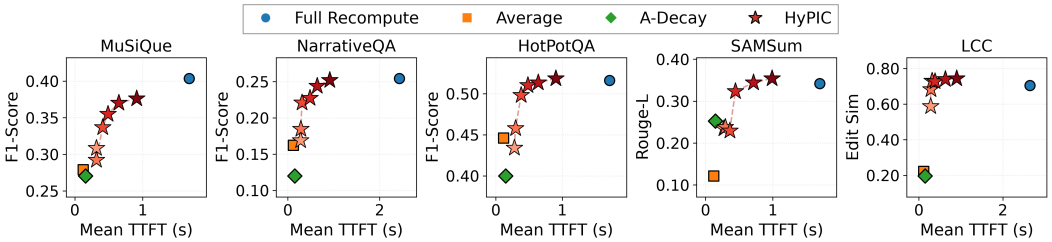


Figure 5: Comparison of HyPIC with baselines on the Qwen3.5-27B model.

because MoE models use fewer active parameters, causing the attention computation to account for a smaller proportion of the total latency, while the bottleneck shifts toward expert routing overhead. As future systems integrate heavily optimized MoE kernels, the relative time spent on attention is expected to increase, thereby amplifying the latency reduction benefits of our method on MoE models.

Robustness. Figure 5 shows that our method exhibits consistent performance trends on the larger Qwen3.5-27B model, confirming its scalability. Notably, TTFT speedup is more pronounced here, as bypassing full recomputation yields greater latency reductions for models with higher computational complexity.

We evaluate the pure Mamba2-based Mamba-Codestral-7B-v0.1 only on code completion using the LCC dataset because it specializes in code. Table 2 demonstrates that our approach achieves the best generation quality. The Average baseline degrades severely as it ignores sequential dependencies between chunks. While A-Decay slightly improves upon Average, it still lags significantly behind the Full Recompute performance. Our method effectively bridges this gap, achieving an optimal trade-off between code quality and speed on pure SSMs.

Comparison with Prefix Caching. We further compare HyPIC with Marconi [29], a prefix-caching method for hybrid LLMs. Following CacheBlend [18], we construct a streaming MuSiQue workload of 750 shuffled requests, with six documents retrieved per query and 150 requests used for warm-up. Across three hybrid models, HyPIC ($K = 64$) achieves a 76.2% cache hit ratio, substantially exceeding Marconi's 18.2%. This advantage becomes even more pronounced in a longer-context setting with 50 retrieved chunks on Qwen3.5-9B ($K = 128$), where HyPIC achieves a 66.76% hit ratio versus only 2.32% for Marconi. These results demonstrate the benefits of PIC reuse in dynamic RAG workloads.

5.3 Ablation Studies

To further validate the effectiveness of each component in our method, we conduct ablation studies on all three models. All experiments are conducted with $K = 64$.

Residual state propagation. We first investigate the impact of the residual state propagation mechanism. As a baseline, we propagate the states directly using A-Decay. As illustrated in Figure 6 ("w/o Residual"), removing this component leads to significant performance degradation across

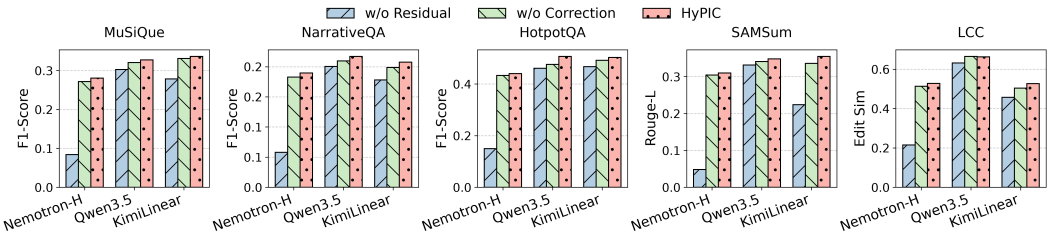


Figure 6: Effectiveness of the residual state propagation and online decay correction strategies.

Table 4: Per-chunk storage breakdown across different models (MB, $K = 64$).

	SSM Cache		KV Cache	
Model	Total	Decays	$L = 512$	$L = 1024$
Nemotron-H-8B	4.190 MB	0.032 MB	2.000 MB	4.000 MB
Qwen3.5-9B	4.100 MB	0.008 MB	2.000 MB	4.000 MB
Kimi-Linear-48B	4.200 MB	0.016 MB	0.580 MB	1.150 MB

all three models. This drop indicates that direct state propagation struggles to preserve long-term historical information, whereas the proposed method can effectively maintain context information.

Online decay correction. To evaluate the online decay correction module, we conduct an ablation study by directly applying the raw offline cumulative decay matrix during the online recomputation phase. As shown in Figure 6 ("w/o Correction"), the online correction mechanism yields consistent performance improvements, demonstrating its effectiveness in mitigating error propagation.

Parallel recomputation. Finally, we investigate the efficiency of the parallel recomputation strategy by replacing it with sequential boundary-token recomputation. As shown in Figure 7, our parallel approach significantly reduces TTFT compared to the sequential baseline, effectively minimizing latency and demonstrating the efficiency of our design.

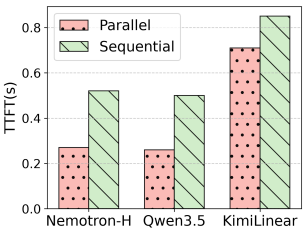


Figure 7: Effectiveness of the parallel recomputation strategy.

5.4 Storage Overhead

We further analyze the storage overhead of HyPIC. For each reusable chunk, HyPIC stores two SSM boundary states, the temporal convolution state, and the associated decay information, while full-attention layers retain standard KV caches. As shown in Table 4, the SSM storage remains constant with respect to chunk length L for a fixed K, whereas KV-cache storage grows linearly with L. Notably, decay information accounts for less than 1% of the total SSM storage, introducing negligible additional overhead. Moreover, all offline caches are offloaded to host memory, while online decoding maintains only a single context-aware SSM state per layer and request.

6 Conclusion

Efficient long-context modeling is crucial for emerging AI applications, yet the heterogeneous nature of hybrid architectures has long hindered effective cache reuse. This paper bridges this gap by introducing HyPIC, a unified PIC reuse framework. HyPIC resolves the structural misalignment between full-attention and SSM layers through a residual state propagation mechanism and prevents cross-chunk state drift via an online decay correction strategy. Moreover, our parallel recomputation pipeline efficiently accelerates state assembly. Delivering a $2\times$ speedup in TTFT while maintaining comparable performance, our method sets a new standard for hybrid model inference.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems*, 2020.

[2] Shervin Minaee, Tomás Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[5] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*, 2023.

[6] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of the European Conference on Artificial Intelligence*, pages 2993–3000, 2025.

[7] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 32779–32798, 2025.

[8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[9] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*, pages 10746–10761, 2025.

[10] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7036–7050, 2024.

[11] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[12] Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

[13] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

[14] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[15] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the Symposium on Operating Systems Principles*, pages 611–626, 2023.

[16] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 37, pages 62557–62583, 2024.

[17] Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Hao Feng, Xusheng Chen, Yizhou Shan, Tao Xie, et al. EPIC: Efficient position-independent caching for serving large language models. In *Proceedings of the International Conference on Machine Learning*, 2025.

[18] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the European Conference on Computer Systems*, pages 94–109, 2025.

[19] Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. In *Proceedings of the ACM on Management of Data*, volume 3, pages 1–28, 2025.

[20] Maciej Pióro, Maciej Wołczyk, Razvan Pascanu, Johannes von Oswald, and João Sacramento. State soup: In-context skill learning, retrieval and mixing. *arXiv preprint arXiv:2406.08423*, 2024.

[21] Tian Yu Liu, Alessandro Achille, Matthew Trager, Aditya Golatkar, Luca Zancato, and Stefano Soatto. Picaso: Permutation-invariant context composition with state space models. *arXiv preprint arXiv:2502.17605*, 2025.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[23] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*, pages 14048–14077, 2023.

[24] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the International Conference on Machine Learning*, pages 10041–10071, 2024.

[25] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of the The International Conference on Learning Representations*, 2025.

[26] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[27] Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

[28] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *ACM Transactions on Computer Systems*, 44(1):1–27, 2025.

[29] Rui Pan, Zhuang Wang, Zhen Jia, Can Karakus, Luca Zancato, Tri Dao, Yida Wang, and Ravi Netravali. Marconi: Prefix caching for the era of hybrid llms. *Proceedings of the Machine Learning and Systems*, 7, 2025.

[30] Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6599–6612, 2025.

[31] Dongyang Ma, Yan Wang, and Lan Tian. Block-attention for efficient prefilling. In *Proceedings of the International Conference on Learning Representations*, 2025.

[32] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[33] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 3119–3137, 2024.

[34] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[35] Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[36] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 70–79, 2019.

[37] Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. Longcoder: A long-range pre-trained language model for code completion. In *Proceedings of the International Conference on Machine Learning*, pages 12098–12107, 2023.

[38] Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ınftybench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 15262–15277, 2024.

[39] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[40] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[41] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063, 2024.

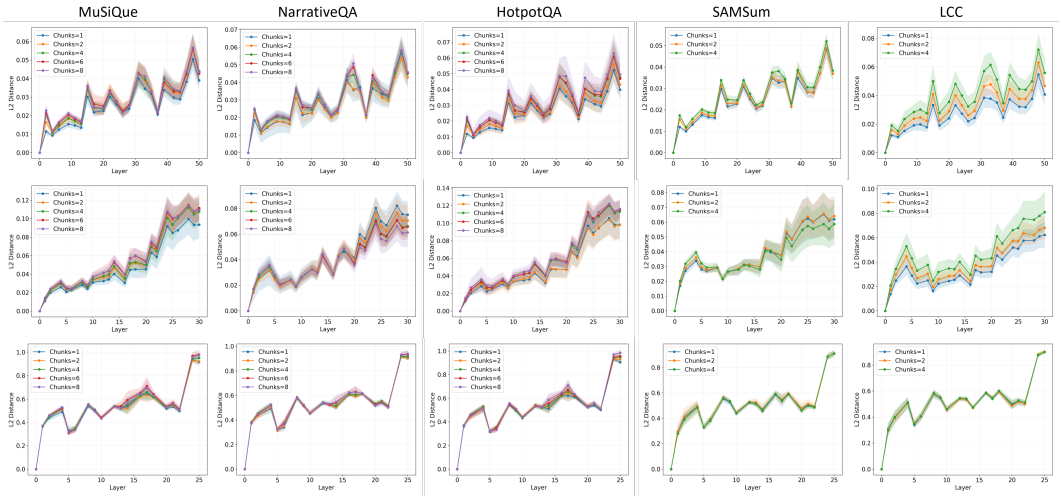


Figure 8: Analysis of the average decay errors in different layers.

A Limitations

While enabling efficient PIC reuse for hybrid architectures, HyPIC presents two limitations. First, evenly selecting K prefix and suffix tokens may be suboptimal, and thus an asymmetric, dataset-specific token distribution could further improve performance. Second, our method requires storing additional offline state for SSM layers. Future work will focus on developing adaptive mechanisms for dynamically allocating boundary tokens and exploring memory-efficient representation methods to further mitigate the additional storage requirements of the offline stage.

B Online Decay Correction

B.1 Analysis of Decay Errors

We conduct a comprehensive analysis of the divergence between decay matrices derived from independent and prefix-based generation. This evaluation is performed across five datasets using three hybrid models. For a given input prompt consisting of N chunks, we obtain the decay matrices using both prefix and independent generation methods.

During prefix generation, we process the entire prompt in a single forward pass to obtain the global decay matrix $\hat{\mathbf{A}}_{1:L}$ at each SSM layer, where $L = \sum_{i=1}^{N} L_i$ is the total length of the N chunks. We then split this matrix along the sequence-length dimension to extract a set of chunk-specific decay matrices, denoted as $\{\hat{\mathbf{A}}^i_{0:L_i}\}$. Conversely, during independent generation, we process each chunk separately, yielding the set of decay matrices $\{\bar{\mathbf{A}}^i_{0:L_i}\}$ at each layer. Finally, we quantify the error of the independently generated decay matrices by computing their L2 distance from the prefix-generated counterparts. The details are described below.

Decay errors across layers. For each prompt, we first compute the L2 distance between the prefix-generated and independently generated decay matrices for each chunk across different layers:

$$s(i) = \|\bar{\mathbf{A}}^i_{0:L_i} - \hat{\mathbf{A}}^i_{0:L_i}\|_2. \tag{8}$$

We then plot the average decay error for each chunk across all datasets. As illustrated in Figure 8, the decay errors vary across different layers and chunks. Consequently, we compute a specific correction term for each layer and each chunk. Moreover, within a given model, the decay errors consistently exhibit similar increasing or decreasing trends. This regularity suggests an underlying structural pattern, which may provide valuable insights for designing more effective correction strategies in future work.

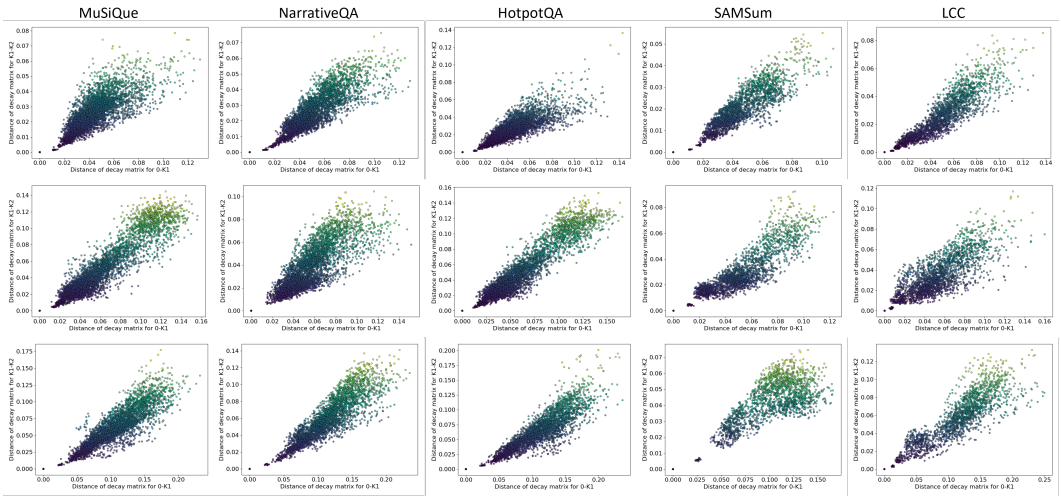


Figure 9: Correlation between the initial and subsequent decay errors.

Correlation between initial and subsequent decay errors. We further investigate the correlation of decay errors between the initial and subsequent portions of the prompt:

$$\begin{aligned} s_{\text{ini}}(i) &= \|\bar{\mathbf{A}}^i_{1:K} - \hat{\mathbf{A}}^i_{1:K}\|_2, \\ s_{\text{sub}}(i) &= \|\bar{\mathbf{A}}^i_{K+1:L_i-K} - \hat{\mathbf{A}}^i_{K+1:L_i-K}\|_2. \end{aligned} \tag{9}$$

We evaluate this relationship by plotting the points $(s_{\text{ini}}(i), s_{\text{sub}}(i))$. As shown in Figure 9, a linear correlation between these two errors is a consistent phenomenon across different datasets and models. Motivated by this observation, we design a correction term based on the initial segment of the prompt to calibrate the subsequent segment.

B.2 Design of the Correction Factor

Motivated by the empirical observations above, we apply an online correction to the subsequent decay matrices. In this section, we provide a detailed justification for our geometric decomposition formulation, demonstrating how it enables orthogonal control over coarse-grained and fine-grained decay deviations.

Multiplicative error in discretization. Modern SSMs map continuous-time parameters to discrete-time matrices. The discrete decay transition matrix is inherently an exponential function of a continuous-time parameter, i.e., $\mathbf{A} = \exp(\Delta \cdot \mathbf{A}_{\text{cont}})$. Consequently, an additive perturbation in the continuous-time domain translates to a multiplicative perturbation in the discrete-time domain: $\exp(\Delta(\mathbf{A}_{\text{cont}}+\delta\mathbf{A})) = \exp(\Delta\mathbf{A}_{\text{cont}})\exp(\Delta\delta\mathbf{A})$. This intrinsic property dictates that the discrepancy between the prefix-generated (online) and independent (offline) decays must be evaluated and corrected multiplicatively.

Coarse-grained global magnitude deviation. While the error is multiplicative, applying it directly as a single high-dimensional vector risks entangling global decay shifts with local semantic changes. Therefore, we first isolate the scalar magnitude m, computed as the L2 norm of the decay matrix, which semantically represents the coarse-grained global retention intensity of the hidden state. An additive error ϵ in the continuous global decay rate yields a discrete magnitude $\hat{m} = m \cdot e^{\epsilon}$. To extract this macro-level shift, we compute the scalar ratio for a given chunk i via direct division:

$$r^i_k = \frac{\hat{m}^i_k}{m^i_k}. \tag{10}$$

This ratio captures the overall exponential scale shift of the decay matrix without being biased by individual channel variations.

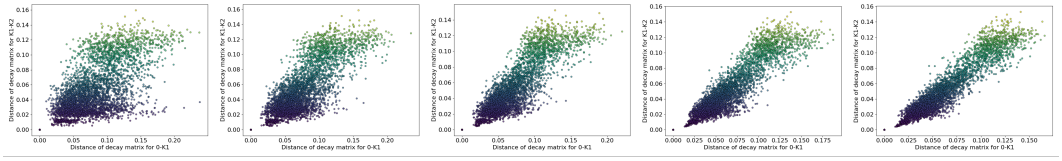


Figure 10: Relationship between the recomputation window size K and the correlation of decay errors with the Qwen3.5-9B model on the HotpotQA dataset.

Fine-grained directional shift. Conversely, the L2-normalized directional vector $\boldsymbol{u}$ represents the fine-grained relative distribution of the decay matrix across different hidden dimensions, determining which specific semantic features are prioritized or forgotten. To quantify this deviation, we perform an elementwise division to define the directional shift:

$$\Delta \boldsymbol{u}_k^i = \frac{\hat{\boldsymbol{u}}_k^i}{\boldsymbol{u}_k^i}. \tag{11}$$

Crucially, because we apply this division to normalized vectors in the exponentiated domain, this operation elegantly factors out the global magnitude scaling. Mathematically, it corresponds to an exact linear subtraction in the continuous logarithmic domain ($\log(\hat{\boldsymbol{u}}_k^i) - \log(\boldsymbol{u}_k^i)$) that cancels out the coarse-grained shift, thereby isolating the fine-grained directional deviation. When applying the correction, we use the Hadamard product ($\odot$) to re-apply this shift, preserving the multiplicative nature of the SSM's forward propagation while maintaining numerical stability.

C Detailed Implementations

C.1 Hyperparameter τ

As illustrated in Figure 10, when the recomputation window K progressively increases, the linear correlation between the decay divergence in the prefix and the subsequent region strengthens significantly. This implies that a small K yields a noisy, high-variance estimation of the decay shift, whereas a larger K provides a robust and highly reliable predictor. Consequently, the correction strength must be adaptively attenuated for small K to prevent the injection of spurious noise into the state propagation.

To achieve this dynamic adaptation, we establish an empirical threshold at 5% of the total chunk length L_i and define a continuous soft-gate $g \in (0, 1)$ as:

$$g = \sigma\left(150.0 \cdot \left(\frac{K}{L_i} - 0.05\right)\right), \tag{12}$$

where $\sigma(\cdot)$ denotes the standard sigmoid function.

This formulation serves as a steep transition mechanism. When the recomputation ratio $\frac{K}{L_i}$ falls below the 5% threshold, the scaling factor 150.0 drastically drives g close to 0, effectively suppressing the correction to avoid noisy updates. Conversely, once the prefix size sufficiently surpasses the threshold, g rapidly saturates to 1.0, allowing the model to fully apply the extracted correction.

Accordingly, the final temporal scaling exponents for magnitude (τ_m) and direction (τ_u) are formulated by multiplying their respective predefined base step sizes, τ_m^{base} and τ_u^{base}, by the gate g:

$$\tau_m = g \cdot \tau_m^{\text{base}}, \quad \tau_u = g \cdot \tau_u^{\text{base}}. \tag{13}$$

In all our experiments, we simply set both τ_m^{base} and τ_u^{base} to 1.0 without any task-specific tuning.

C.2 Execution Pipeline

Algorithm 1 outlines the online execution pipeline designed for hybrid architectures. During the inference forward pass, the input prompt is processed layer by layer. Given the architectural heterogeneity

Algorithm 1 Online Recomputation Pipeline for Hybrid Architectures

Require: Input states $\mathbf{H}$, Boundary index set $\mathcal{I}$, Offline KV/SSM caches

Retrieve M matching chunks and compute position offsets $\{s_i\}_{i=1}^{M}$
for layer $l = 1$ to N **do**
  if Layer l is **Full-Attention then**
    Load $\{(\mathbf{K}_i, \mathbf{V}_i)\}_{i=1}^{M}$ and concatenate to obtain $(\mathbf{K}, \mathbf{V})$
    if Model utilizes RoPE **then**
      $\mathbf{k}'_t \leftarrow \mathbf{R}_{s_i}\mathbf{k}_t$
    end if
    Gather $\mathbf{E} = \mathbf{H}[\mathcal{I}]$, compute $\mathbf{Q}^s, \mathbf{K}^s, \mathbf{V}^s$, and overwrite $(\mathbf{K}, \mathbf{V})$ to yield $\mathbf{K}^u, \mathbf{V}^u$
    $\mathbf{O}^{(l)} \leftarrow \mathrm{softmax}\left(\frac{\mathbf{Q}^s(\mathbf{K}^u)^T}{\sqrt{d}}\right)\mathbf{V}^u$
  else if Layer l is **SSM then**
    Load reference caches: $\{\mathbf{S}^i_K, \mathbf{S}^i_{L-K}, \bar{\mathbf{A}}^i_{1:K}, \mathbf{D}^i_{K:L-K}\}_{i=1}^{M}$
    Step 1: Initial Zero-State Scan
    Extract $2K$-length cross-boundary windows: $\mathbf{H}_{\text{bndy}} \leftarrow \{\mathbf{H}^i_{[-K:K]}\}_{i=1}^{M}$
    $\mathbf{U}_K, \hat{\mathbf{D}}_{-K:K}, \hat{\mathbf{A}}_{1:K} \leftarrow \text{SSM}_{\text{scan}}(\mathbf{H}_{\text{bndy}}, \mathbf{0})$
    Step 2: State Propagation & Correction
    Initialize starting anchor $\hat{\mathbf{S}}^0_{L-K}$ from preceding historical context
    for $i = 1$ to M **do**
      $\hat{\mathbf{S}}^i_K \leftarrow \mathbf{U}^i_K + \hat{\mathbf{D}}^i_{-K:K}\hat{\mathbf{S}}^{i-1}_{L-K}$ ▷ Eq. 7
      $\bar{r}^i \leftarrow \mathbb{E}\left[\frac{\hat{m}^i_k}{m^i_k}\right], \quad \Delta\bar{\boldsymbol{u}}^i \leftarrow \mathbb{E}\left[\frac{\hat{\boldsymbol{u}}^i_k}{\boldsymbol{u}^i_k}\right]$
      $\hat{\mathbf{D}}^i_{K:L-K} \leftarrow m_{\text{target}}(\bar{r}^i)^{\tau_m}\,\mathrm{Norm}\left(\boldsymbol{u}_{\text{target}} \odot (\Delta\bar{\boldsymbol{u}}^i)^{\tau_u}\right)$ ▷ Eq. 5
      $\hat{\mathbf{S}}^i_{L-K} \leftarrow \mathbf{S}^i_{L-K} + \hat{\mathbf{D}}^i_{K:L-K}\left(\hat{\mathbf{S}}^i_K - \mathbf{S}^i_K\right)$ ▷ Eq. 6
    end for
    Step 3: Final Parallel Scan (Boundary Output Generation)
    $\mathbf{S}_{\text{init}} \leftarrow \mathrm{stack}\left(\{\hat{\mathbf{S}}^{i-1}_{L-K}\}_{i=1}^{M}\right)$
    $\mathbf{O}^{(l)}_{\text{bndy}} \leftarrow \text{SSM}_{\text{scan}}(\mathbf{H}_{\text{bndy}}, \mathbf{S}_{\text{init}})$
    Overwrite $\mathbf{H}[\mathcal{I}]$ with $\mathbf{O}^{(l)}_{\text{bndy}}$
  end if
end for
return Final output states $\mathbf{O}^{(N)}$

of hybrid models, the cached states and input features must be routed through distinct operational branches depending on the specific layer type. Below, we elaborate on the specific recomputation operations tailored for both full-attention and SSM layers.

Pipeline in full-attention layers. For full-attention layers, we adopt a pipeline similar to other recomputation-based PIC methods [18, 17]. During the offline generation stage, we generate and store the KV caches for each independent chunk. During the online recomputation stage, a retrieval process identifies reusable states based on the input prompt, forming a sequence of M matching KV cache chunks, denoted as $\{(\mathbf{K}_i, \mathbf{V}_i)\}_{i=1}^{M}$. These retrieved chunks are then loaded and concatenated to construct the stacked KV cache $(\mathbf{K}, \mathbf{V})$.

Subsequently, for models utilizing rotary positional encodings (e.g., Qwen3.5), a position re-encoding step is applied to the stacked KV caches. This process primarily targets the Rotary Positional Encoding (RoPE) [41]. Given a retrieved independent KV cache chunk with an original position span of $[0, \ldots, l-1]$, we must re-encode its positions to $[s, \ldots, s+l-1]$, where s is the new starting position of this chunk within the concatenated prompt. Because RoPE's angle addition property allows linear shifting, we can efficiently achieve this position re-encoding by applying an additional rotation with an offset of s to the existing cache. Therefore, given the original key vector $\mathbf{k}_t$ of the t-th token in the chunk, its re-encoded version $\mathbf{k}'_t$ is computed as:

$$\mathbf{k}'_t = \mathbf{R}_s\mathbf{k}_t, \tag{14}$$

where $\mathbf{R}_s$ is the block-diagonal rotation matrix with an offset of s:

$$\mathbf{R}_s = \mathrm{diag}\left(\mathbf{R}_s^{(0)}, \mathbf{R}_s^{(1)}, \ldots, \mathbf{R}_s^{(\frac{d}{2}-1)}\right). \tag{15}$$

Here, d is the hidden dimension of the key vector, and each 2×2 block $\mathbf{R}_s^{(j)}$ is defined as:

$$\mathbf{R}_s^{(j)} = \begin{bmatrix} \cos(s\theta_j) & -\sin(s\theta_j) \\ \sin(s\theta_j) & \cos(s\theta_j) \end{bmatrix}, \quad \theta_j = 10000^{-2j/d}. \tag{16}$$

Finally, the KV caches for the selected boundary tokens are updated by recomputing their key-value pairs using the full continuous context [18]. Specifically, the input embeddings for the boundary tokens, $\mathbf{E} \in \mathbb{R}^{n \times d}$, are gathered from the full input sequence according to the boundary index set $\mathcal{I}$. The new queries $\mathbf{Q}^s$, keys $\mathbf{K}^s$, and values $\mathbf{V}^s$ are then computed from $\mathbf{E}$. These newly computed $\mathbf{K}^s$ and $\mathbf{V}^s$ overwrite the corresponding entries in the stacked KV cache, yielding the fully updated caches $\mathbf{K}^u$ and $\mathbf{V}^u$. The final attention output is then computed as:

$$\mathbf{O} = \mathrm{softmax}\left(\frac{\mathbf{Q}^s(\mathbf{K}^u)^T}{\sqrt{d}}\right)\mathbf{V}^u. \tag{17}$$

Pipeline in SSM layers. In SSM layers, modern SSMs typically maintain two distinct states to capture both long-term and short-term context. While the core methodology discussed in our main paper primarily focuses on the long-term memory state, we also account for the short-term memory state, which corresponds to the state generated after applying a causal convolution over a fixed window (e.g., 4 tokens).

Consequently, during the offline generation phase, we cache both the short-term and long-term memory states for each chunk at two specific positions: immediately after the K-th token and after the $(L-K)$-th token. Additionally, we store the stepwise decay matrices for the initial K tokens, alongside the cumulative decay matrix spanning from token K to $L-K$. During the online recomputation phase, matching chunks are retrieved using the same strategy as in the full-attention layers. The cached states of all retrieved chunks are then loaded and fed into our proposed recomputation and propagation pipeline. Ultimately, the final short-term state is directly extracted from the recomputation results of the final K tokens in the last chunk.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction in this paper accurately reflect the paper's contributions in enabling position-independent cache reuse for hybrid models.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are included in the Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not contain any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We introduce the details of the experiment, which includes the information of in Section 5.1 and Appendix C.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data and code are uploaded to GitHub.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details of the experimental settings in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Throughout our experiments, we keep the temperature at 0 and use greedy decoding to strictly control the reproducibility of the generated outputs. To reliably evaluate efficiency, the reported TTFT is averaged over five runs. We also provide error bars in Figure 8 for a more rigorous decay error analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5.1 provides detailed information about the computational resources used in our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and acknowledge the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work contributes to the position-independent cache reuse for hybrid models. It does not impact society at large.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code, data, models utilized in this paper are cited and credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components. We only utilize it for editing and formatting.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.