HORIZONKV: AN OVERLAP-AWARE SPARSE ATTENTION SYSTEM FOR LONG-HORIZON AUTOREGRESSIVE VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive diffusion transformers (AR-DiTs) generate video chunk by chunk and can in principle extend generation to arbitrary video length. However, as KV caches grow, existing models are forced to retain only a short window of local frames, causing catastrophic forgetting and drift. Retrieval-based sparse attention retrieves important historical frames, but GPU capacity limits the range of accessible history. We observe that access to the full KV history substantially improves generation quality. However, realizing this benefit raises two challenges. The first is *insufficient memory capacity*. Recent work addresses this by offloading KV to the CPU, but retrieval overhead slows generation. The second is *increased compute FLOPs*, as longer histories contain more important KV entries and thus require more attention FLOPs. To address these challenges, we present HorizonKV, a training-free sparse attention system for long-horizon autoregressive video diffusion. To address insufficient memory capacity, we introduce *Overlap-Aware Execution*, which uses tiered storage and selectively overlaps KV transfers with attention computation to hide transfer latency. To address increased compute FLOPs, we introduce *Residency-Aware Sparse Attention*, which adaptively selects two-level sparsity configurations based on KV cache residency on the GPU. It maintains quality while hiding as much communication overhead as possible without increasing FLOPs. Experiments show HorizonKV advances the Pareto frontier over all baselines, with up to 1.486× end-to-end speedups. Our source code is available at the URL.

1 INTRODUCTION

Video generation is moving toward applications that demand several minutes of interactive and continuous video, such as world models, interactive narratives, and game-like rollouts (Wan et al., 2025; Kong et al., 2024; Valevski et al., 2025). Diffusion transformers (DiTs) (Peebles & Xie, 2023) use joint spatial and temporal attention to produce high-quality videos, but generate all frames at once (Yang et al., 2025b; Wu et al., 2025). Their attention cost scales quadratically with video length (Vaswani et al., 2017; Xia et al.), making long video generation computationally expensive and limiting interactive use (Xia et al., 2025a) (Figure 1a). The emerging autoregressive diffusion transformer (AR-DiT) paradigm sidesteps this limitation (Yin et al., 2025; Huang et al., 2026; Xiao et al., 2024; Yang et al., 2025a) by enabling incremental generation. As shown in Figure 1b, AR-DiTs generate a video chunk by chunk and cache the keys and values of previously generated frames, supporting streaming and interactive video generation (Xia et al., 2026).

In AR-DiTs, historical KV plays a key role in generation quality by providing context from previously generated frames. A common approach to using this history effectively is retrieval-based attention (Hu et al., 2026; Zhao et al., 2026). These methods retrieve the historical KV entries most relevant to the current chunk and use them as attention context. This improves generation quality while reducing computation compared with dense attention over the full history, making retrieval-based attention a promising approach to long-video generation.

More importantly, we find that ***expanding the retrieval range improves generation quality*** because important information in long videos is often scattered across distant frames (§3-Obs.1). Under

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

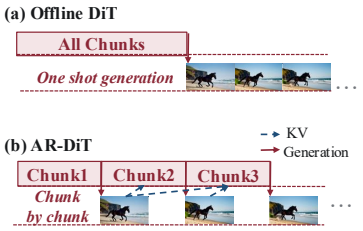


Figure 1: Comparison between offline DiT and AR-DiT.

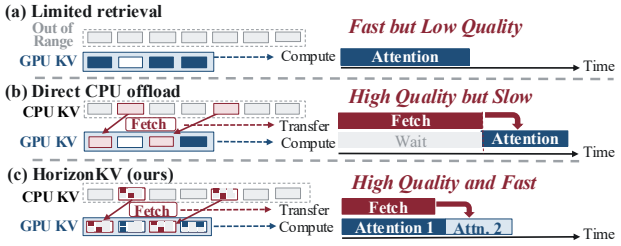


Figure 2: Effects of KV retrieval range and execution strategy on generation quality and latency.

a fixed compute budget, a narrow retrieval range can therefore exclude useful context and limit generation quality (Figure 2a, §3-Obs.1). We thus aim to enable retrieval from the full history. However, as the generated videos become longer, achieving this goal raises two key challenges.

Challenge 1 (C1). Insufficient memory capacity. Long-video generation produces large KV caches, and GPU memory cannot hold the full history (§3-Obs.2) (Gao et al., 2025; Zhang et al., 2026). For a 1.3B backbone, an 80 GB GPU cannot even store the KV cache for one minute of 480p video. Recent work explores offloading KV directly to the CPU (Zhao et al., 2026), but these transfers remain on the critical path, as shown in Figure 2b. Attention must wait for these transfers, slowing generation and restricting real-time and interactive use.

Challenge 2 (C2). Increased compute FLOPs. We observe that preserving generation quality requires more historical KV Cache as video length increases, since relevant context spans more frames (§3-Obs.3). Processing these additional KV entries therefore increases the computational cost of attention. This also exacerbates C1's transfer bottleneck as retrieval demand grows.

To address these challenges, we propose HorizonKV, a training-free system for long-horizon autoregressive video diffusion. By drawing on the full history through overlap-aware sparse attention, it improves long-video quality while accelerating generation (Figure 2c). To address **C1**, HorizonKV presents *Overlap-Aware Execution (OAE)*, which places historical KV between CPU and GPU memory by access recency and selectively applies split attention when a cost comparison favors it, hiding communication behind computation. To address **C2**, HorizonKV introduces *Residency-Aware Sparse Attention (RASA)*, which uses per-call KV cache residency to select two-level sparsity configurations within a fixed compute budget, favoring high quality and low transfer volume. This keeps transfer overhead low enough for *Overlap-Aware Execution* to hide it.

Our contributions are as follows.

- We characterize KV caches in AR-DiTs and show how interactions among KV Cache retrieval range, sparsity patterns, and transfer costs affect generation quality and speed. Based on these findings, we develop a training-free sparse attention system that supports long-horizon generation with AR-DiTs and can be easily applied to other base models.
- We introduce two coupled mechanisms, *Overlap-Aware Execution* and *Residency-Aware Sparse Attention*. Under quality constraints, they use KV residency to coordinate tiered storage, selective overlap, and compute budget allocation for sparse attention. Together, they improve quality through retrieval from the full history while accelerating generation.
- Across two representative open-source AR-DiTs, HorizonKV advances the Pareto frontier for quality and latency in long-video generation, substantially improving quality while achieving end-to-end speedups up to 1.486×.

2 Background and Related Work

Autoregressive Video Diffusion. Standard video diffusion transformers (DiTs) iteratively denoise latents using bidirectional attention. The predicted clean latents are $\widehat{\mathbf{x}}_0 = f_\theta(\mathbf{x}_t, t, \mathbf{c})$, where $\mathbf{x}_t$ denotes noisy latents at timestep t and $\mathbf{c}$ denotes conditioning inputs. AR-DiTs instead generate videos chunk by chunk, denoising each chunk conditioned on cached keys and values from previously generated frames (Huang et al., 2026; Yang et al., 2025a; Yin et al., 2025). To bound memory

usage, models commonly retain initial frames as attention sinks alongside a sliding window of the latest W chunks (Huang et al., 2026; Yang et al., 2025a). Thus, this sink-plus-local cache supports indefinite generation in principle, but excludes earlier frames outside the retained context.

Recent work explores complementary ways to preserve or compress long-range history in autoregressive video diffusion. Forcing-KV applies hybrid KV cache compression with structured and similarity-based pruning (Ji et al., 2026). MemFlow maintains adaptive memory for consistent long video narratives (Ji et al., 2025). Adaptive caching methods further reuse historical computation to reduce the cost of video diffusion inference (Kahatapitiya et al., 2025). These methods reduce or reuse historical state, whereas HorizonKV keeps the full history available in host memory and dynamically retrieves the required KV entries (Chen et al., 2026c; Chu et al., 2025).

Sparse Attention in Autoregressive Video Diffusion. Sparse attention in AR-DiTs operates at two granularities, *frame-wise* and *token-wise*. For each head, we denote queries, keys, and values by Q, K, V, and the head dimension by d_h. The attention weights are $A = \mathrm{softmax}(QK^\mathsf{T}/\sqrt{d_h})$.

Frame-wise sparsity restricts attention to a small set of frames selected by importance. We denote the selected frame set by S and the frame containing key j by $f(j)$. This selection defines the binary mask $P_{ij}^{\mathrm{frame}} = \mathbf{1}[f(j) \in S]$. Representative retrieval methods include LongLive-RAG (Hu et al., 2026) and OmniMem (Zhao et al., 2026). LongLive-RAG retrieves historical latents using learned embeddings and uses Window Temporal Delta Loss to reduce local redundancy. OmniMem uses adaptive window exclusion to mitigate local bias and query-shared selection with per-head access to limit memory overhead. Recent adaptive memory methods also study long-range historical information for maintaining consistency in extended video generation (Ji et al., 2025).

Token-wise sparsity preserves important query-key token pairs in the attention map. For efficiency, selection and execution typically operate at block granularity (Xi et al., 2025; Zhang et al., 2025b). The binary mask P_{ij}^{token} specifies whether the pair (i, j) is retained. LightForcing, for instance, combines frame selection with block-level sparsity (Lv et al., 2026). Other video diffusion methods use adaptive sparse patterns, trainable sparse attention, semantic-aware token permutation, or online block selection to improve the hardware efficiency of sparse attention (Xia et al., 2025b; Yang et al., 2026; Zhang et al., 2025a; Sun et al., 2026; Yuan et al., 2025; Gu et al., 2026; Chen et al., 2026b).

Together, the two granularities define hierarchical sparsity, $P = P^{\mathrm{frame}} \odot P^{\mathrm{token}}$. The first level selects important frames, and the second selects token pairs within those frames. A configuration $c = (k, \rho)$ selects k frames and retains a fraction ρ of the block pairs within each selected frame. The resulting sparse attention output is

$$O_P = \mathrm{softmax}\left(QK^\mathsf{T}/\sqrt{d_h} + \log P\right) V, \qquad \log 0 = -\infty. \tag{1}$$

Attention recall $\mathrm{Recall}(P) = \frac{\sum_{i,j} A_{ij} P_{ij}}{\sum_{i,j} A_{ij}}$ measures the fraction of attention mass retained at either selection level (Jiang et al., 2024). It is the fraction of dense attention mass retained by the sparse mask. At a fixed budget, higher recall means fewer important interactions are discarded and generally indicates better fidelity (Chen et al., 2026a).

Offloaded Attention in Language Models. Selective retrieval from host memory is well established in language-model inference, with representative methods including ShadowKV (Sun et al., 2024) and InfiniGen (Lee et al., 2024). These methods target single-query decoding, where memory traffic dominates the cost per step. This makes indexing, speculation, or reconstruction attractive. Efficient memory management and hierarchical KV storage have also been studied for large language-model serving (Sheng et al., 2023; Kwon et al., 2023). In contrast, AR-DiTs process thousands of queries against a shared history at each denoising step and are compute-bound (Xi et al., 2025; Xia et al., 2025b). Per-query scoring costs are therefore harder to amortize. However, the substantial attention workload also offers an opportunity to hide communication overhead behind computation. This opportunity to overlap communication and computation underpins our system.

3 Motivating Observations

In this section, we present key observations on the memory characteristics of AR-DiTs and the opportunities that motivate our work. For this analysis, we generate 60s 480p video using the 1.3B Self Forcing AR-DiT (Huang et al., 2026) on an A100 GPU, following the setup in Appendix A.

Observation 1. ***Longer retrievable history improves generation quality.*** In Figure 3a, we vary only the retrievable KV range while keeping the attention budget (the number of selected frames) fixed. Expanding this range improves generation quality. Figure 3b shows the attention distribution averaged across all heads when generating the chunk at 60 seconds, which explains this improvement (Wen et al., 2025; Chen et al., 2026b). Sink and local frames account for 8% and 19% of the attention weight, respectively. The remaining 73% is concentrated in a few narrow peaks within the intermediate history. These peak locations depend on content rather than recency. The attention maps in Figure 3c further show that distant memories receive substantial attention weight. Because important memories are both distant and irregularly distributed, restricting retrieval to a short window can exclude useful context and reduce generation quality.

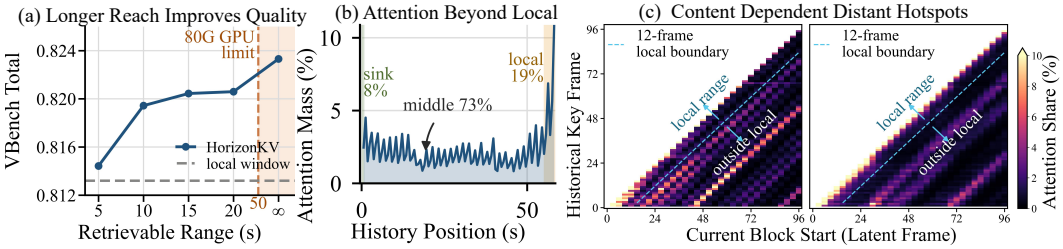


Figure 3: Long-range history improves video generation quality. (a) shows that the total VBench (Huang et al., 2024) score increases as the retrieval range expands. (b) shows the distribution of attention weight across sink, local, and intermediate frames, with intermediate frames receiving most of the nonlocal weight. (c) shows distant attention hotspots beyond the local window, with locations varying across the first 96 latent frames.

Observation 2. ***Full history exceeds GPU capacity, requiring CPU offloading with transfer costs that cannot always be hidden.*** Figure 4a shows that this 1.3B model's KV cache grows rapidly. After accounting for other memory allocations, an 80 GB GPU can store roughly 50 seconds of history. Retaining longer histories therefore requires offloading to host memory (Zhao et al., 2026; Sun et al., 2024; Lee et al., 2024). Figure 4b further analyzes a generation run using frame-wise sparse attention. It compares attention computation on selected GPU-resident KV Cache with transfers of selected CPU-resident KV Cache back to the GPU. Some attention computation can overlap with these transfers, but some transfer overhead remains exposed.

Observation 3. ***Longer history requires more KV frames, and hierarchical sparsity offers multiple high-quality configurations under a fixed budget.*** Figure 5a shows that frame-wise sparse attention requires selecting more frames to maintain generation quality as video length increases. Under a fixed FLOP budget, a natural approach is to use two-level sparsity, reducing within-frame density to expand frame coverage. Figure 5b compares configurations under the same budget B. Selecting only a few dense frames provides insufficient historical context and therefore yields lower quality. Reducing within-frame density expands frame coverage, improving quality until it reaches a plateau. However, allocating the same budget across too many frames leaves too little signal in each selected frame and reduces quality. Several configurations $c = (k, \rho)$ (§2) therefore yield similarly high quality. Let $\mathcal{C}_B$ denote the offline candidate set under this budget and $g(c)$ the mean generation quality measured with fixed configuration c. We define $g_{\min} = \max_{c \in \mathcal{C}_B} g(c) - \delta$, where δ denotes the quality tolerance. The candidate set on the quality plateau is $\mathcal{C}_Q = \{c \in \mathcal{C}_B \mid g(c) \geq g_{\min}\}$. It includes configurations whose calibrated scores are no more than δ below the best score.

Insight 1. Combine full-history storage with overlapped execution (Obs. 1, 2). Retrieval from the full KV history can substantially improve quality, but the history does not fit in GPU memory. Distributing KV across host and device memory incurs transfer overhead, part of which can be hidden by computation. However, some transfer latency remains exposed, so the execution engine must also handle calls with incomplete overlap.

Insight 2. Adapt hierarchical sparsity within a high-quality plateau based on compute and transfer times (Obs. 2, 3). Hierarchical sparsity improves quality within the same compute budget. Frame count and block density determine both transfer volume and the computation available to hide transfer overhead. This allows quality to guide budget allocation before execution. Configurations are prioritized when computation on resident KV can hide their transfer overhead.

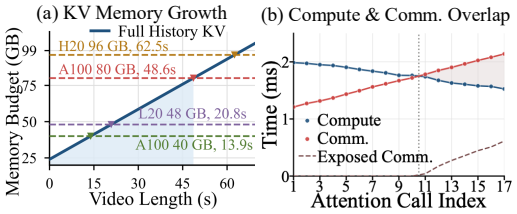


Figure 4: The full KV history becomes increasingly costly to store and retrieve as generation continues. (a) shows memory usage growing with video length and exceeding typical device memory capacity. (b) shows attention computation and KV communication times for individual attention calls, with computation on resident KV hiding communication until an exposed transfer tail emerges.

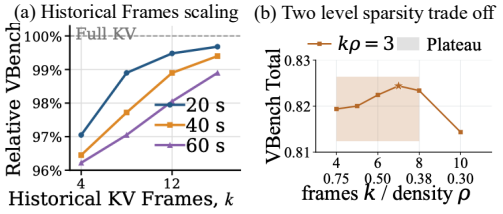


Figure 5: Two-level sparsity balances history coverage and computation. (a) Longer videos require more KV frames to maintain quality. (b) Under fixed budget, reducing the retained fraction within each frame expands coverage and improves quality until a plateau, while spreading budget across too many frames leaves insufficient signal and lowers quality. The curves use $c = (k, \rho)$, and shaded regions mark near optimal plateaus.

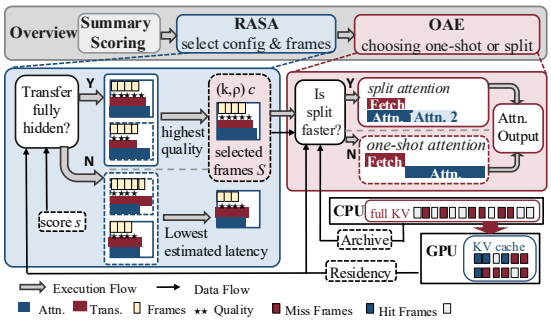


Figure 6: Overview of HorizonKV.

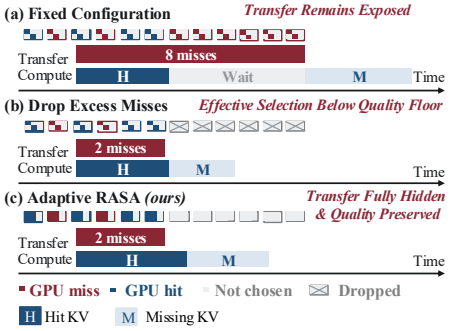


Figure 7: Effectiveness of RASA.

4 HorizonKV

Motivated by these insights, we propose HorizonKV, an overlap-aware sparse attention system for long-horizon autoregressive video diffusion. Figure 6 gives a system overview. The full KV history resides in host memory, while a bounded GPU cache stores frequently accessed KV entries. For each new frame f, we mean-pool its keys into a GPU-resident summary, $\bar{k}_f = \frac{1}{N_f} \sum_{j \in f} k_j$, where N_f is the token count in frame f. We omit layer and head indices throughout.

Attention computation proceeds in three steps. (1) We use Equation 2 to score each historical KV summary against current queries, where N_q denotes the query count.

$$\bar{q} = \frac{1}{N_q} \sum_i q_i, \qquad s_f = \frac{\bar{q}^\top \bar{k}_f}{\sqrt{d_h}}, \quad f = 1, \ldots, F. \tag{2}$$

Here, F denotes the total number of historical frames. (2) Using the scores s_f and the residency of all KV entries, *Residency-Aware Sparse Attention* (RASA) chooses a sparsity configuration $c = (k, \rho)$ together with the corresponding frame set S. (3) *Overlap-Aware Execution* (OAE) retrieves KV for frames in S and computes attention under configuration c, selecting one-shot or split attention to hide communication overhead behind computation whenever possible (Ye et al., 2025). We first describe how OAE executes attention and selects its execution mode in §4.1, and then explain how RASA selects the sparsity configuration in §4.2.

4.1 Overlap-Aware Execution

As shown in Figure 6, OAE takes the selected frame set S and sparsity configuration c as input, fetches the corresponding KV entries, and computes attention. OAE supports two execution modes, *one-shot attention* and *split attention*. When KV transfer costs justify the splitting overhead, OAE

Algorithm 1 Overlap-Aware Execution

Input Queries Q, selected frames S, configuration c, resident frames R
Output Attention output O
1: $H \leftarrow S \cap R$, $M \leftarrow S \setminus R$
2: Estimate $T_{\text{one}}, T_{\text{split}}$ using Eq. 4
3: **if** $M = \varnothing$ or $T_{\text{one}} \leq T_{\text{split}}$ **then**
4: Fetch KV for M from host memory
5: **return** the output of $\text{Attn}_c(Q, S)$
6: **end if**
7: $(O_H, \ell_H) \leftarrow \text{Attn}_c(Q, H)$ **concurrently with** fetching KV for M
8: After transfers complete, compute $(O_M, \ell_M) \leftarrow \text{Attn}_c(Q, M)$
9: **return** O from Eq. 3

Algorithm 2 Residency-Aware Sparse Attention

Input Frame scores $(s_f)_{f=1}^{F}$, resident frames R, candidates $\mathcal{C}_Q$, calibrated quality g
Output Selected frames S and configuration c
1: $\mathcal{F} \leftarrow \varnothing$
2: Rank frames per head and scan residency
3: **for each** $c = (k, \rho) \in \mathcal{C}_Q$
4: $S(c) \leftarrow$ top-k frames from each ranking
5: Estimate candidate costs using Eq. 4
6: **if** c satisfies Eq. 6, $\mathcal{F} \leftarrow \mathcal{F} \cup \{c\}$
7: **if** $\mathcal{F} \neq \varnothing$, select $c \in \mathcal{F}$ with highest $g(c)$ Break ties by fewer misses, then smaller k
8: **else** select $c \in \mathcal{C}_Q$ with smallest $T(c)$
9: **return** $(S(c), c)$

executes attention in two passes, overlapping transfers with attention computation on resident KV. Otherwise, splitting introduces more overhead than it saves, so OAE uses *one-shot attention*, fetching all missing KV entries before computing attention. Once a frame is fully generated, OAE stores its KV entries in host memory and adds them to the GPU cache, managed using LRU replacement.

Split attention. Let S denote the selected frames and R the frames with GPU-resident KV. OAE splits S into hits $H = S \cap R$ and misses $M = S \setminus R$ (Alg. 1, line 1). Configuration c defines the combined sparsity mask P_c. For a frame set X, $\text{Attn}_c(Q, X)$ applies this mask to K_X, V_X using Equation 1. It returns O_X and the row-wise log normalizer ℓ_X. OAE overlaps attention over H with transfers of KV for M from host memory (Alg. 1, line 7). Once transfers finish, it computes attention over M (Alg. 1, line 8). It then merges the two outputs row by row (Alg. 1, line 9) using

$$\ell = \text{logaddexp}(\ell_H, \ell_M), \qquad O = e^{\ell_H - \ell} O_H + e^{\ell_M - \ell} O_M. \tag{3}$$

Execution policy. We select the execution mode using timing models obtained through profiling (NVIDIA, 2026). Let x denote the estimated transfer time for misses. The estimated attention times are c_H for resident KV, c_M for missing KV, and c_U for one-shot attention. The overhead e accounts for additional kernel launches and output merging. OAE compares the two estimated execution times (Alg. 1, line 2), given by

$$T_{\text{one}} = x + c_U, \qquad T_{\text{split}} = \max(x, c_H) + c_M + e. \tag{4}$$

Splitting is faster when overlap savings exceed added compute, launch, and merge costs. Equivalently,

$$T_{\text{split}} < T_{\text{one}} \iff \min(x, c_H) > c_H + c_M + e - c_U. \tag{5}$$

If there are no misses or one-shot attention is no slower, OAE first fetches any missing KV and then returns the attention output over S (Alg. 1, lines 3 to 5). Otherwise, OAE uses split attention. Even when $x > c_H$, split attention can reduce call latency, although $(x - c_H)_+$ of the transfer time remains exposed. Fully hiding transfer latency requires $x \leq c_H$. Therefore, the selected configuration's computation time on resident KV, c_H, defines its transfer-hiding budget. RASA aims to keep x within this budget by selecting from the quality plateau candidate set $\mathcal{C}_Q$ (§4.2).

4.2 Residency-Aware Sparse Attention

We adaptively select the sparsity configuration c and the corresponding frame set S using calibrated quality estimates to improve overlap between communication and computation. Each configuration $c = (k, \rho)$ specifies how many frames are selected and their within-frame density.

Why residency-aware selection matters. Multiple configurations satisfy the same budget $B = k\rho$. Observation 3 shows that several configurations achieve similar quality, but require different amounts of data transfer. Figure 7a illustrates a limitation of using a fixed configuration. Selecting many frames at low density creates many misses but provides little computation on resident KV to hide transfers. Discarding excess misses (Figure 7b) loses selected information and can produce a configuration outside the quality plateau. Therefore, we select configurations within the fixed budget and quality plateau based on actual misses and estimated computation times. For example, selecting fewer frames by reducing k and increasing their density ρ can reduce transfers and potentially increase resident computation, improving overlap (Figure 7c).

Selection policy. We measure the FLOP budget B in full-frame equivalents, so $c = (k, \rho)$ costs $k\rho$ in these units. We keep the budget fixed using $\rho = B/k$. Offline calibration provides $\mathcal{C}_Q$ and $g(c)$ (§3). Appendix E evaluates transferability and RASA's quality on held-out prompts.

RASA starts with $\mathcal{F} = \varnothing$ (Alg. 2, line 1). It ranks historical frames by s_f independently for each head and uses the same ranking for all candidates (Alg. 2, lines 2 to 4). For each candidate $c = (k, \rho)$, we define $S(c) = \mathrm{TopK}\Big((s_f)_{f=1}^{F}, k\Big)$, $H(c) = S(c) \cap R$, $M(c) = S(c) \setminus R$, where TopK returns the indices of the k highest-scoring frames for each head. A single prefix scan provides miss counts for all nested candidate sets. Each miss transfers its entire layer-head-frame slice regardless of ρ, while computation uses only retained block pairs. OAE's timing models estimate $x(c)$, $c_H(c)$, and both execution times (Alg. 2, line 5). These times vary at fixed FLOPs. RASA forms the feasible set under OAE's execution policy (Alg. 2, line 6):

$$\begin{aligned} T(c) &= \min\{T_{\text{one}}(c), T_{\text{split}}(c)\}, \\ \mathcal{F} &= \big\{c \in \mathcal{C}_Q \mid M(c) = \varnothing \ \vee \ \big[x(c) \le c_H(c) \ \wedge \ T_{\text{split}}(c) < T_{\text{one}}(c)\big]\big\}. \end{aligned} \tag{6}$$

The final configuration and frame sets are

$$c = \begin{cases} \arg\max\limits_{c' \in \mathcal{F}} g(c'), & \mathcal{F} \neq \varnothing, \\ \arg\min\limits_{c' \in \mathcal{C}_Q} T(c'), & \mathcal{F} = \varnothing, \end{cases} \qquad S = S(c).$$

Here, c' denotes a candidate and c the selected configuration. Thus, RASA maximizes calibrated quality among candidates with no predicted exposed transfers under OAE (Alg. 2, line 7). Ties favor fewer misses and then smaller k. If $\mathcal{F}$ is empty, it minimizes estimated latency within $\mathcal{C}_Q$ (Alg. 2, line 8). It passes c and S to OAE (Alg. 2, line 9).

Default hyperparameters. We set the default total attention FLOP budget to $B = 3$ full-frame equivalents and allocate it uniformly across frames selected by importance. The quality floor is set 0.01 VBench points below the highest calibrated score. For all evaluated backbones, $\mathcal{C}_Q$ consists of configurations with $k \in \{3, \ldots, 12\}$ and $\rho = 3/k$. Calibrated quality peaks at $k = 7$. Detailed settings and implementations are provided in Appendix A and Appendix E.

5 Evaluation

5.1 Experimental Setup

Models. We evaluate two representative open-source models, Self Forcing (Huang et al., 2026) and LongLive (Yang et al., 2025a). For both models, we generate 2 min videos at 480p using their default settings. We also evaluate scalability with video lengths of 1-3 min in §5.5.

Datasets and metrics. Following previous work, we evaluate generation quality using VBench-Long and its official dataset (Huang et al., 2024). We report quality, semantic, and standard aggregate scores. Results for all sixteen dimensions are reported in Appendix B. To illustrate the tradeoff between quality and speed, our main comparison includes a Pareto curve of quality versus latency. We evaluate efficiency using a single 80 GB A100 GPU (NVIDIA, 2022), with PCIe transfers between the GPU and pinned host memory (NVIDIA, 2026).

Baselines. We compare HorizonKV with state-of-the-art (SOTA) *Token-wise* and *Frame-wise* sparse attention methods. (1) *LightForcing* (Lv et al., 2026) combines historical frame retrieval with *Token-wise* sparse attention and adjusts sparsity across generated chunks. (2) *LongLive-RAG* (Hu et al., 2026) retrieves historical frames per chunk and supports CPU offloading. (3) *OmniMem* (Zhao et al., 2026) retrieves historical KV blocks per query group and head, with CPU offloading for memory management. To isolate the effect of historical range, (4) *HorizonKV-GPU* disables offloading and limits retrieval candidates to the 190 frames that fit in GPU memory. All other settings use the default settings, detailed in Appendix A. We also evaluate two ablation baselines to isolate the contributions of the two components. *HorizonKV-wo-OAE* retains the selected configuration and frame set S, but performs KV transfers serially before computing attention over the union. This isolates the effect of overlap. *HorizonKV-wo-RASA* fixes k and $\rho = B/k$ across calls to assess how residency-aware adaptation affects speed and quality.

Table 1: Quality and efficiency comparison for 120s videos. Latency in seconds.

Method	LongLive					Self Forcing				
	Quality ↑	Semantic ↑	Total ↑	Speedup ↑	Latency ↓	Quality ↑	Semantic ↑	Total ↑	Speedup ↑	Latency ↓
Base model	0.8126	0.6344	0.7769	1.000×	181.157	0.8354	0.6426	0.7968	1.000×	181.092
Light Forcing	0.8130	0.6288	0.7762	1.376×	131.700	0.8346	0.6426	0.7962	1.373×	131.895
LongLive-RAG	0.8143	0.6411	0.7796	0.570×	317.820	0.8315	0.6839	0.8020	0.570×	317.705
OmniMem	0.8264	0.6397	0.7891	0.159×	1136.421	0.8442	0.6710	0.8096	0.162×	1115.510
HorizonKV-GPU	0.8204	**0.6844**	0.7932	**1.515**×	**119.542**	0.8486	0.6936	0.8176	**1.521**×	**119.078**
HorizonKV (**ours**)	**0.8300**	0.6833	**0.8007**	1.485×	121.974	**0.8544**	**0.7066**	**0.8248**	1.486×	121.865

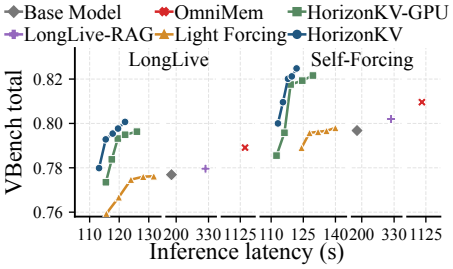


Figure 8: Quality versus inference latency for LongLive and Self Forcing. Lines connect the points on each method's Pareto frontier.

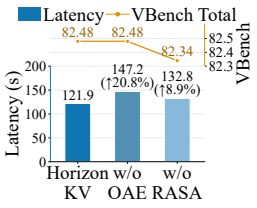


Figure 9: Effects of removing OAE or fixing RASA configurations on VBench quality and inference latency.

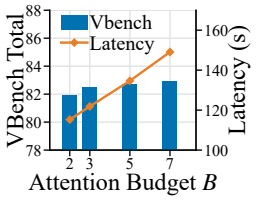


Figure 10: Attention budget sensitivity. Larger budgets yield modest quality gains with near-linear latency growth.

5.2 Main Results

Table 1 reports quality and speedup under default configurations, and Figure 8 shows the corresponding Pareto frontiers. Since LongLive-RAG and OmniMem do not target acceleration, we report their default results without Pareto curves. HorizonKV achieves the best tradeoff between quality and latency, with total VBench gains of 0.0238 to 0.0280 and up to $1.486\times$ speedups.

Different limitations keep the baselines from reaching this frontier. *LightForcing* accelerates generation through sparsity, trading slight quality losses for reduced computation. *LongLive-RAG* searches accumulated history using the previous latent and keeps its selection fixed across denoising steps. This may not reflect changing attention patterns, while latent encoding adds overhead. *OmniMem* caps the selected block count, yet transfers entire chunks. As history grows, the union of selected chunks across queries exceeds GPU cache capacity, forcing repeated transfer and attention passes that substantially slow generation. HorizonKV uses RASA to adaptively expand historical coverage with less computation, improving quality, while OAE hides transfer latency. *HorizonKV-GPU* uses HorizonKV's selection, sparsity, and execution strategies but restricts retrieval to GPU-resident frames, limiting generation quality. Appendix D complements these results with visual comparisons.

5.3 Ablations

Figure 9 isolates the contribution of each component described in §4 using Self Forcing.

Without OAE. We retain RASA's selected configuration and frame set S, but complete KV transfers before computing *one-shot* attention. OAE achieves a $1.21\times$ speedup. This gain comes from overlapping transfers with attention computation on resident KV, reducing exposed transfer latency.

Without RASA. We retain OAE but fix $c = \arg\max_{c' \in \mathcal{C}_Q} g(c')$, the configuration maximizing calibrated quality, rather than adapting it. RASA maintains nearly identical quality while achieving a $1.09\times$ speedup. By adapting frame count and within-frame density, RASA better aligns transfer and resident computation times, enabling OAE to hide transfer latency exposed by the fixed choice.

5.4 Case Study

Time Breakdown. Figure 11a compares attention execution times for Self Forcing, separating preprocessing (e.g., latent encoding, importance scoring, and configuration selection), attention computation, exposed transfer latency, and postprocessing (e.g., output merging and branch fusion).

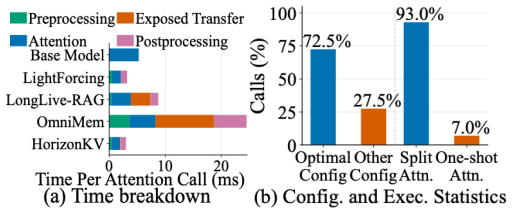


Figure 11: Case study of execution cost and runtime decisions. (a) Time breakdown across methods. (b) Shares of configuration choices and attention execution modes on Self Forcing.

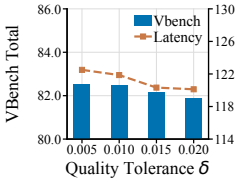


Figure 12: Quality and latency across different quality-floor tolerance settings.

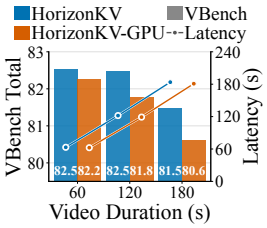


Figure 13: Quality and latency on Self Forcing at video lengths of 60, 120, and 180 seconds

HorizonKV's lightweight preprocessing selects a compact subset of KV entries, reducing computation while preserving quality. OAE overlaps transfers with attention on resident KV, hiding nearly all transfer latency. Other methods incur higher attention computation costs or exposed transfer latency, resulting in longer execution times.

Configuration and Execution Statistics. Figure 11b reports RASA's configuration choices and OAE's execution modes on Self Forcing when offloading is required. Most calls select the configuration that maximizes $g(c)$ over $\mathcal{C}_Q$, while the rest use other plateau configurations. RASA usually selects the highest-scoring calibrated configuration. OAE predominantly uses split attention, highlighting the importance of overlapping transfers with computation on resident KV.

5.5 Sensitivity and Scalability

Budget Sensitivity. Figure 10 reports results for $B \in \{2, 3, 5, 7\}$ when generating 120-second 480p videos using Self Forcing. Increasing the budget modestly improves quality while increasing generation latency approximately linearly.

Quality-Floor Sensitivity. Under the same setting, Figure 12 compares results for $\delta \in \{0.005, 0.01, 0.015, 0.02\}$. Performance remains stable around the default $\delta = 0.01$. Setting $\delta = 0.015$ slightly improves speed, while $\delta = 0.005$ slightly improves quality at similar latency. At $\delta = 0.02$, quality drops more noticeably without further speedup.

Scalability. Figure 13 compares HorizonKV and HorizonKV-GPU using Self Forcing to generate 480p videos lasting 1, 2, and 3 minutes. HorizonKV's quality advantage grows with video duration as GPU capacity limits the baseline's accessible history. Meanwhile, HorizonKV's generation time per second of video remains nearly constant.

6 Conclusion and Discussion

We present HorizonKV, a training-free sparse attention system for long-horizon autoregressive video diffusion. It addresses GPU memory constraints and growing attention costs to enable efficient retrieval from the full history. Overlap-Aware Execution combines tiered KV storage with selective overlap between transfers and resident computation. Residency-Aware Sparse Attention adapts hierarchical sparsity to actual KV residency under a fixed compute budget. Together, these mechanisms preserve quality while reducing exposed transfer costs. Experiments on Self Forcing and LongLive demonstrate improved generation quality and faster inference with a bounded GPU cache.

Future work could extend HorizonKV to longer videos and more advanced retrieval. Retaining the full history increases host memory usage by about 1.15 GB per second of video, motivating compression or eviction to secondary storage. Moreover, our current implementation uses mean-pooled top-k selection without a specialized keyframe selector. As a general execution system, HorizonKV is complementary to retrieval design and could incorporate specialized methods such as LongLive-RAG and OmniMem to further improve the selection of relevant historical context at inference.

REPRODUCIBILITY STATEMENT

HorizonKV is a training-free inference system evaluated on the open-source Self Forcing and LongLive models. The evaluation protocol, benchmark, metrics, and hardware setup are described in Section 5, with model settings and baseline configurations provided in Appendix A. Appendix E details the cost calibration, KV transfer granularity, sparse attention block allocation, cache synchronization, and exact merging of attention outputs used in our implementation. We evaluate generation quality using VBench-Long and report detailed quality dimensions in Appendix B. Efficiency measurements are conducted on a single NVIDIA A100 80 GB GPU with PCIe transfers between GPU memory and pinned host memory. Ablation and sensitivity studies further characterize the effects of execution overlap, adaptive sparsity, attention budgets, and quality tolerance.

ETHICS STATEMENT

HorizonKV improves the efficiency and visual quality of long-horizon video generation while reducing GPU memory requirements, potentially supporting applications in creative production, education, and interactive simulation. These improvements may also lower the cost of producing longer and more detailed synthetic videos for harmful purposes, including impersonation, misinformation, and non-consensual content. Although HorizonKV focuses on inference optimization without additional model training, it does not address biases or harmful generation capabilities inherited from the underlying models. We encourage responsible deployment with appropriate content safeguards, provenance tracking, and clear disclosure of AI-generated media. Applications involving identifiable individuals should respect their privacy and consent, and deployments should follow the usage restrictions of the underlying models.

AI USE STATEMENT

We used AI tools to assist with language polishing and code debugging during the preparation of this work. For the manuscript, AI assistance focused on improving grammar, clarity, and readability. For the code, AI tools assisted in identifying and resolving implementation errors.

REFERENCES

Pengtao Chen, Xianfang Zeng, Maosen Zhao, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2957–2965, 2026a.

Ruichen Chen, Keith Mills, Liyao Jiang, Chao Gao, and Di Niu. Re-ttention: Ultra sparse visual generation via attention statistical reshape. *Advances in Neural Information Processing Systems*, 38:58029–58055, 2026b.

Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, et al. Longlive-2.0: An nvfp4 parallel infrastructure for long video generation. *arXiv preprint arXiv:2605.18739*, 2026c.

Huanpeng Chu, Wei Wu, Guanyu Feng, and Yutao Zhang. Omnicache: A trajectory-oriented global perspective on training-free cache reuse for diffusion transformer models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16302–16312. IEEE, 2025.

Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, et al. Seerattention-r: Sparse attention adaptation for long reasoning. *arXiv preprint arXiv:2506.08889*, 2025.

Youping Gu, Xiaolong Li, Yuhao Hu, Chen Minqi, and Bohan Zhuang. Blade: Block-sparse attention meets step distillation for efficient video generation. In *International Conference on Learning Representations*, volume 2026, pp. 56021–56045, 2026.

Qixin Hu, Shuai Yang, Wei Huang, Song Han, and Yukang Chen. Longlive-rag: A general retrieval-augmented framework for long video generation. *arXiv preprint arXiv:2606.02553*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Sihui Ji, Xi Chen, Shuai Yang, Xin Tao, Pengfei Wan, and Hengshuang Zhao. Memflow: Flowing adaptive memory for consistent and efficient long video narratives. *arXiv preprint arXiv:2512.14699*, 2025.

Yicheng Ji, Zhizhou Zhong, Jun Zhang, Qin Yang, XiTai Jin, Ying Qin, Wenhan Luo, Shuiyang Mao, Wei Liu, and Huan Li. Forcing-kv: Hybrid kv cache compression for efficient autoregressive video diffusion models. *arXiv preprint arXiv:2605.09681*, 2026.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15240–15252. IEEE, 2025.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

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 29th symposium on operating systems principles*, pp. 611–626, 2023.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 155–172, 2024.

Chengtao Lv, Yumeng Shi, Yushi Huang, Ruihao Gong, Shen Ren, and Wenya Wang. Light forcing: Accelerating autoregressive video diffusion via sparse attention. *arXiv preprint arXiv:2602.04789*, 2026.

NVIDIA. NVIDIA A100 80GB PCIe GPU: Product brief, March 2022. URL https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/PB-10577-001_v02.pdf.

NVIDIA. CUDA C++ best practices guide, 2026. URL https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/. Last updated 2026-01-08.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

Wenhao Sun, Rong-Cheng Tu, Yifu Ding, Jingyi Liao, Zhao Jin, Shunyu Liu, and Dacheng Tao. Vorta: Efficient video diffusion via routing sparse attention. *Advances in Neural Information Processing Systems*, 38:7837–7863, 2026.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, volume 2025, pp. 73754–73776, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yuxin Wen, Jim Wu, Ajay Jain, Tom Goldstein, and Ashwinee Panda. Analysis of attention in video diffusion transformers. *arXiv preprint arXiv:2504.10317*, 2025.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Yifei Xia, Fangcheng Fu, Hao Yuan, Suhan Ling, Xupeng Miao, Huixia Li, Yuxi Ren, Xin Xia, Xuefeng Xiao, and Bin CUI. Echoattention: Exploiting token-pair redundancy and frame-block similarity for efficient video generation. In *Forty-third International Conference on Machine Learning*.

Yifei Xia, Fangcheng Fu, Hao Yuan, Hanke Zhang, Xupeng Miao, Yijun Liu, Suhan Ling, Jie Jiang, and Bin Cui. Tridentserve: A stage-level serving system for diffusion pipelines. *arXiv preprint arXiv:2510.02838*, 2025a.

Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15982–15993. IEEE, 2025b.

Yifei Xia, Hao Yuan, Suhan Ling, Haoran Sun, Hanke Zhang, Xupeng Miao, Fangcheng Fu, and Bin Cui. Adaptive resource management and quality control for streaming video generation. *arXiv preprint arXiv:2606.15319*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025a.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *Advances in Neural Information Processing Systems*, 38:96965–96991, 2026.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025b.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, et al. Flashinfer: Efficient and customizable attention engine for llm inference serving. *Proceedings of Machine Learning and Systems*, 7, 2025.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974. IEEE, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, et al. Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse–linear attention. In *International Conference on Learning Representations*, volume 2026, pp. 138288–138305, 2026.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. *arXiv preprint arXiv:2502.04507*, 2025b.

Lin Zhao, Yushu Wu, Yifan Gong, Yanzhi Wang, and Pu Zhao. Omnimem: Scalable and adaptive memory retrieval for long video generation. *arXiv preprint arXiv:2605.30519*, 2026.

Table 2: Self Forcing per-dimension values for 120-second videos.

Dimension	Base	LF	RAG	Omni	GPU	Horizon
Subject Consistency	96.81	96.45	98.24	97.51	97.63	97.69
Background Consistency	95.94	95.37	96.56	95.72	96.62	96.62
Temporal Flickering	98.90	98.83	99.47	98.79	98.91	99.00
Motion Smoothness	97.82	98.28	98.68	97.64	98.07	97.76
Dynamic Degree	68.10	62.60	51.01	69.69	66.88	79.38
Aesthetic Quality	57.52	59.56	59.33	63.04	62.62	61.52
Imaging Quality	70.25	70.29	67.47	70.08	71.54	70.94
Object Class	74.79	83.04	84.46	86.63	87.78	88.89
Multiple Objects	55.99	48.67	70.24	45.31	64.00	60.92
Human Action	72.02	68.75	73.21	79.69	75.00	75.00
Color	85.17	91.13	88.09	92.95	92.68	94.47
Spatial Relationship	69.16	74.94	80.92	73.25	76.49	82.69
Scene	23.71	17.94	22.50	19.85	23.89	28.72
Appearance Style	20.64	20.70	20.51	20.37	20.29	20.23
Temporal Style	20.95	20.83	21.58	23.70	23.35	23.48
Overall Consistency	22.78	21.97	21.91	23.88	23.33	23.20

A Experimental Setup

Base model. We evaluate the Wan-based 1.3B checkpoints of Self Forcing and LongLive at 480×832 and 16 fps, with KV caches stored in BF16. We use the default settings of both base models. Each latent frame contains 1560 tokens. Each chunk consists of three latent frames and follows the checkpoint's four-step denoising schedule. Both models use 12-frame windows, with 1 sink in Self Forcing and 3 sinks in LongLive.

LightForcing. We separately adapt both backbones using official procedures, configurations, and datasets. We set `keep_frames=3`, `keep_sink=1`, `keep_near=1`, `sparsity=0.85`, and `sparsity_base=0.95`.

LongLive-RAG. We evaluate LongLive-RAG on LongLive and Self Forcing using the officially released backbone checkpoints and retrieval encoder. We retain the learned retrieval encoder, retrieve six latent blocks, and keep the five-block recency exclusion fixed. Both indexing and encoding costs are included.

OmniMem. We separately post-train LongLive and Self Forcing following the paper's recipe with the official code and data, retaining selection, compression, per-head access, and CPU offloading. Each backbone uses its corresponding post-trained model and gate weights with the original inference pipeline.

HorizonKV-GPU. We keep all KV entries within a truncated historical range on the GPU. For 480p, 120-second generation on an A100 80 GB, we budget 14 GB for model weights and reserve another 10 GB for activations, workspaces, summaries, and temporary buffers to provide headroom against out-of-memory errors. This leaves approximately $C_{\text{history}} = 56$ GB for historical KV. Each latent frame requires $v_f = 0.28754$ GB across the full model. The resulting capacity is $\lfloor C_{\text{history}}/v_f \rfloor = 194$ latent frames. Considering the memory fluctuations introduced by the actual VAE stages and other factors, we adopt a window size of 190 latent frames, covering 39.6% of the 480 latent frames in a 120-second video. All other selection and sparsity settings remain unchanged.

HorizonKV. We use the same GPU memory budget as HorizonKV-GPU, allocating approximately 56 GB to historical KV and limiting the GPU cache to 190 latent-frame equivalents. The cache uses LRU replacement, while the full history remains available in host memory.

B VBench Dimensions

Tables 2 and 3 report all sixteen VBench dimensions for the Self Forcing and LongLive comparisons, respectively. Per-dimension scores are multiplied by 100.

Table 3: LongLive per-dimension values for 120-second videos.

Dimension	Base	LF	RAG	Omni	GPU	Horizon
Subject Consistency	97.95	96.45	98.46	98.82	98.82	98.74
Background Consistency	95.63	95.25	95.79	97.76	97.08	97.76
Temporal Flickering	99.19	98.25	99.19	99.36	99.16	99.21
Motion Smoothness	98.88	98.28	98.99	98.41	98.64	98.44
Dynamic Degree	29.86	40.14	26.25	50.83	30.83	35.97
Aesthetic Quality	58.41	59.66	59.44	58.45	60.54	61.83
Imaging Quality	68.32	69.05	69.00	64.06	68.73	70.81
Object Class	74.90	70.95	73.35	59.14	76.44	81.09
Multiple Objects	64.38	57.19	78.72	68.34	70.78	65.90
Human Action	58.33	59.03	49.31	63.48	61.11	65.28
Color	71.48	83.88	87.86	89.54	89.54	82.15
Spatial Relationship	88.37	81.04	82.45	79.20	90.27	86.09
Scene	14.87	15.85	9.77	23.20	25.84	30.10
Appearance Style	18.85	18.77	19.20	19.93	19.24	19.93
Temporal Style	22.67	23.92	22.53	19.64	23.10	22.89
Overall Consistency	24.45	23.02	23.44	23.36	23.89	23.76

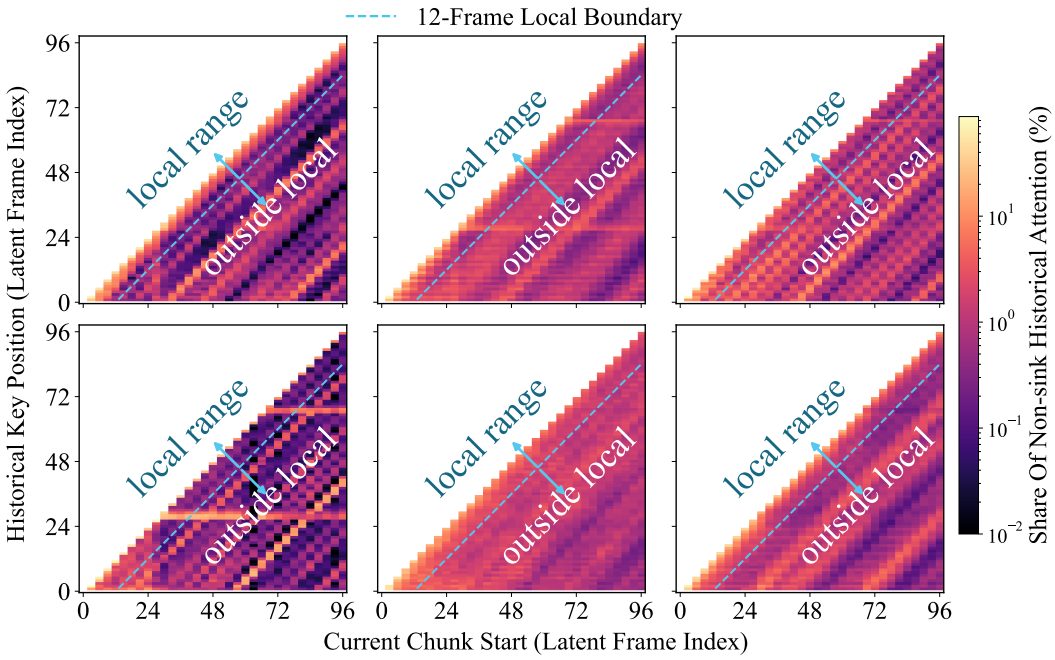


Figure 14: Attention maps across generation chunks for six heads, showing historical attention beyond the local window.

C Examples of Long-Video Attention Maps

Figure 14 illustrates several heads across the first 96 latent frames of a 120-second video. Attention mass does not simply decay with time, but appears in a few irregular intermediate-history peaks whose locations follow content. These displayed heads do not have a pronounced sink or local preference. This is the phenomenon behind Observation 1 and explains why a fixed nearby window can omit important evidence and why retrieval should cover the full history.

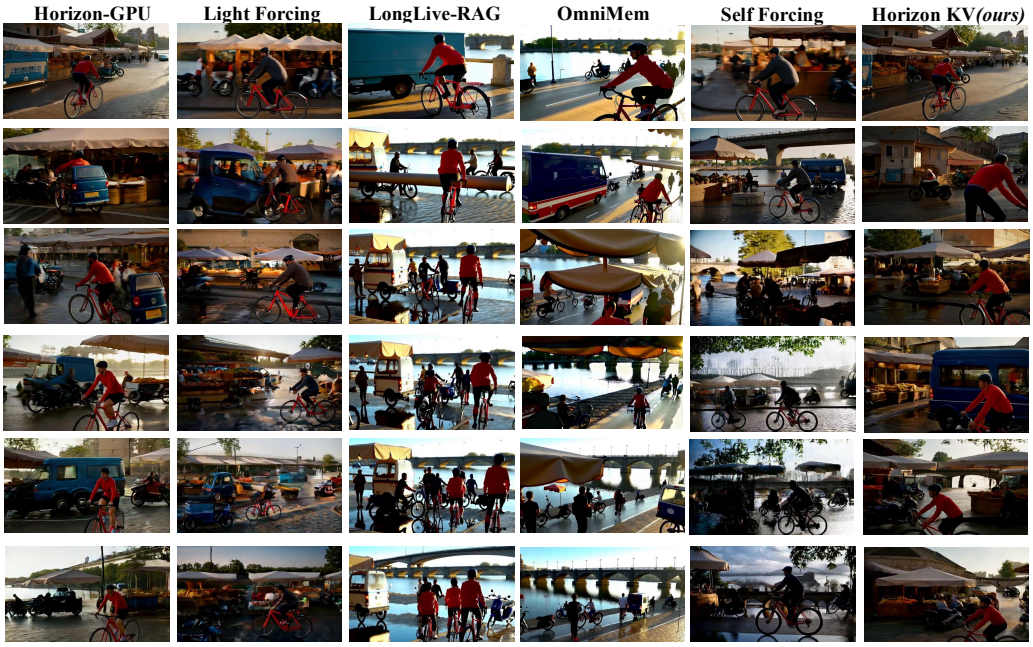


Figure 15: **Prompt:** *A single continuous street documentary shot follows a red bicycle courier accelerating through a bustling riverside market at late afternoon. The cyclist weaves carefully past fruit stalls, pedestrians, parked scooters, and a blue delivery van while fabric awnings flutter overhead and warm sunlight flickers across the wet cobblestones. The camera tracks from a low side angle, moves ahead to face the cyclist, then rises into a smooth overhead view as the route opens toward a stone bridge and the river beyond. Realistic balance, wheel rotation, shadows, reflections, coherent spatial relationships, natural crowd motion, no cuts, no text, no logos.*

D Visual Examples

We organize some snapshot comparisons from the main experiment in Figure 15, Figure 16 and Figure 17. The results show that, compared with other methods, our approach achieves better visual quality with fewer artifacts. In contrast, other methods are prone to visual degradation and drift when generating long videos.

E Implementation Details

Cost Quantities. The runtime reads cache state at each layer call before eviction. For head h, selected missing and resident frames are M_h and H_h. A frame slice occupies $v_h = 2N_f d_h b$ bytes. If $\mathcal{B}_{H,h}$ contains resident query-key block pairs for head h, then $F_H = 4d_h \sum_h \sum_{(u,v)\in\mathcal{B}_{H,h}} n_u n_v$. A simple calibrated model is

$$V_M = \sum_h |M_h| v_h, \qquad x = a_x \mathbf{1}[V_M > 0] + V_M/\beta, \qquad c_H = a_H + F_H/\eta_H. \tag{7}$$

Here β is effective transfer bandwidth and η_H shape-dependent attention throughput. Fixed terms contain dispatch and mask work. A more detailed lookup table can replace this model without changing Algorithm 2. Concurrent paths need separate calibration because DMA, memory bandwidth, and occupancy can interact. Complete hiding is predicted from these calibrated quantities rather than guaranteed by hardware.

Transfer Granularity and Implementation. A miss fetches the entire selected layer-head-frame slice, with transfer bytes counted over valid tokens. This makes candidate bytes known before mask construction and avoids query-dependent host block gathering. Split execution overlaps DMA

transfers with resident block scoring, mask construction, and attention computation. The estimates c_H, c_M, c_U include block scoring, mask construction, and attention.

Token-wise Sparse Implementation. We use 64-token blocks and mask padded positions. The sizes n_u, n_v in F_H denote padded block sizes, both equal to 64. With $m = \lceil N_f/64 \rceil$ blocks per frame, each query block retains Bm key blocks across k frames. We assign $\lfloor Bm/k \rfloor$ blocks per frame and distribute the remainder in descending s_f order before splitting S into H and M. Here, $\rho = B/k$ is the average retained fraction, and uniform allocation is subject to integer rounding. For each head, block importance is $r_{uv} = \tilde{q}_u^\top \tilde{k}_v / \sqrt{d_h}$, where $\tilde{q}_u$ and $\tilde{k}_v$ are means over valid tokens in query block u and key block v. Each query block selects the highest-scoring key blocks within each frame according to its quota. Resident block scoring therefore requires no missing KV.

Profile. We profile full-frame-slice transfers, packing, resident and miss kernels, one-shot kernels, and output merging separately, rather than assuming $c_U = c_H + c_M$. This accounts for differences in kernel efficiency and the additional overhead of split execution. Concurrent copy and attention are also measured separately to capture resource contention that isolated timings may miss. These measurements calibrate the execution costs used by OAE and RASA.

Calibration Robustness and Validation. We assess plateau robustness on Self Forcing at the default budget $B = 3$ using 120-second 480p videos. We divide the prompt suite into two disjoint, equally sized sets $\mathcal{D}_1$ and $\mathcal{D}_2$, balancing coverage across evaluation dimensions. The split and generation seeds are fixed. Both sets use the same candidate configurations and VBench-Long evaluation protocol.

For each set $\mathcal{D}_i$, we evaluate every configuration $c \in \mathcal{C}_B$ with that configuration fixed throughout generation. Let $g_i(c)$ denote its aggregate score on $\mathcal{D}_i$. Using the same tolerance $\delta = 0.01$ and deterministic tie-breaking rule, we define

$$c_i^\star \in \arg\max_{c \in \mathcal{C}_B} g_i(c), \qquad \mathcal{C}_Q^{(i)} = \{c \in \mathcal{C}_B \mid g_i(c) \geq g_i(c_i^\star) - \delta\}.$$

The two curves exhibit similar plateau trends and identify the same plateau configurations and highest-scoring configuration,

$$\mathcal{C}_Q^{(1)} = \mathcal{C}_Q^{(2)}, \qquad c_1^\star = c_2^\star.$$

We freeze both calibration results before running RASA on each prompt set. Let G_{ij} denote the aggregate score on $\mathcal{D}_j$ when RASA uses g_i and $\mathcal{C}_Q^{(i)}$. Runs with $i \neq j$ assess transfer to prompts excluded from the corresponding calibration, whereas runs with $i = j$ provide references on the calibration prompts. On each prompt set, both policies use identical prompt-seed pairs and unchanged execution settings. We observe

$$|G_{11} - G_{12}| \approx 0.006, \qquad |G_{21} - G_{22}| \approx 0.005,$$
$$|G_{11} - G_{21}| \approx 0.003, \qquad |G_{12} - G_{22}| \approx 0.004.$$

The first row quantifies stability across prompt sets under each frozen calibration, while the second row quantifies sensitivity to the calibration set on identical evaluation prompts. Both comparisons show small differences in aggregate scores. The matching plateau sets and maxima, together with these results across prompt sets and calibration choices, support empirical robustness across the two sampled subsets. They do not establish robustness to arbitrary distribution shifts or guarantee quality for individual prompts.

Exact Merge. For a fixed query row, each disjoint pass has unnormalized weight $u_i = e^{\ell_i}$ and normalized output O_i. The union numerator is $u_H O_H + u_M O_M$ and denominator is $u_H + u_M$, giving Equation 3. Stable logaddexp avoids exponentiating large raw logits. Empty misses return the resident output directly. Entries cannot be overwritten before reader completion. One-shot and split execution use identical per-frame quotas, importance scores, and tie-breaking rules, so both produce the same mask.

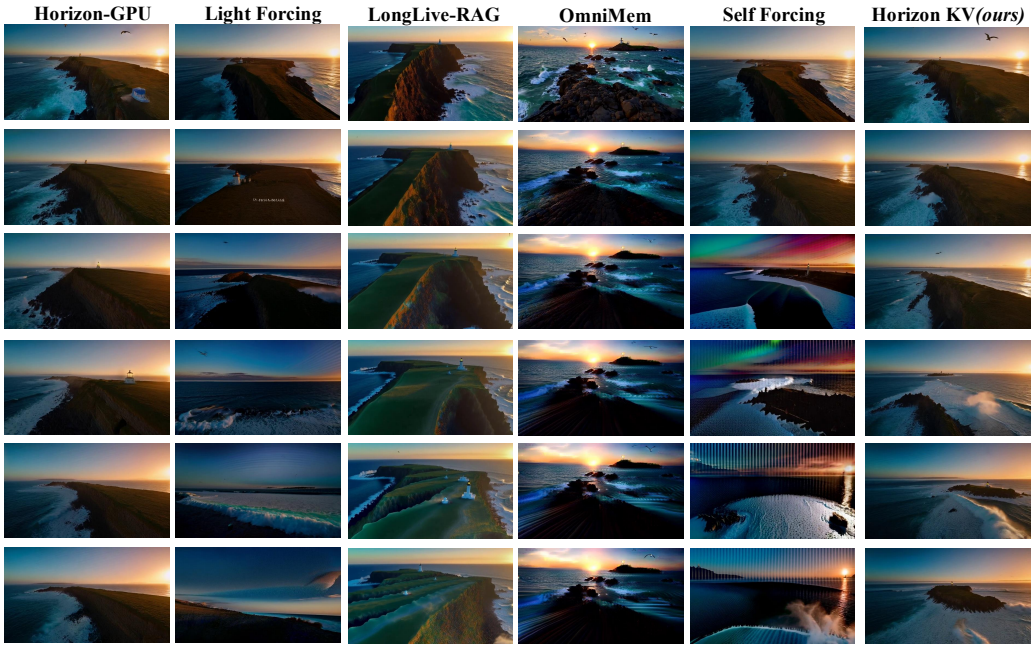


Figure 16: **Prompt:** *A single continuous cinematic drone shot follows a rugged Pacific coastline at golden hour: turquoise waves break against dark sea cliffs, seabirds sweep through the wind, and a small lighthouse on a distant island begins to glow as the sun sinks toward the horizon. The camera glides forward along the cliff edge, slowly descends toward the surf, and ends above the wet rocks as the sky changes from warm orange to deep blue twilight. Photorealistic natural textures, stable geometry, gentle realistic motion, rich atmospheric light, subtle ocean mist, no cuts, no text, no logos.*

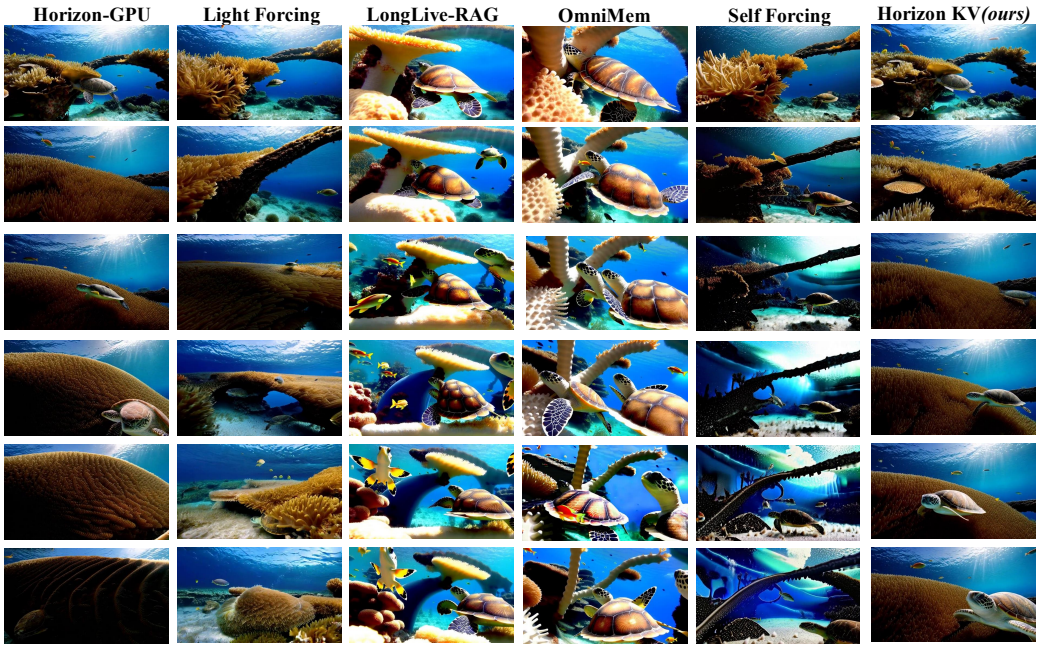


Figure 17: **Prompt:** *A single continuous underwater nature documentary shot drifts through a vibrant coral reef at midday. Schools of turquoise and orange fish weave around branching coral, a sea turtle passes close to the lens, suspended particles sparkle in shafts of sunlight, and a gentle current makes sea grass and soft corals sway. The camera slowly circles a reef arch, follows the turtle into a blue opening, and returns to the original reef as sunlight changes subtly with passing clouds. Photorealistic marine life, coherent colors and anatomy, smooth buoyant camera movement, no cuts, no text, no logos.*