HORIZONKV: THREE-STATE KV CACHE COMPRESSION UNDER A SHRINKING GENERATION HORIZON FOR VISUAL AUTOREGRESSIVE MODELING

Anonymous authors
Paper under double-blind review



Figure 1: **Extreme-budget generation with HorizonKV. Top:** Full Cache of Infinity-8B (Han et al., 2025) with 85 GB memory; **bottom:** HorizonKV with 0.85 GB at 1% historical KV storage, where the generation global structure and major visual details remain close to the Full Cache.

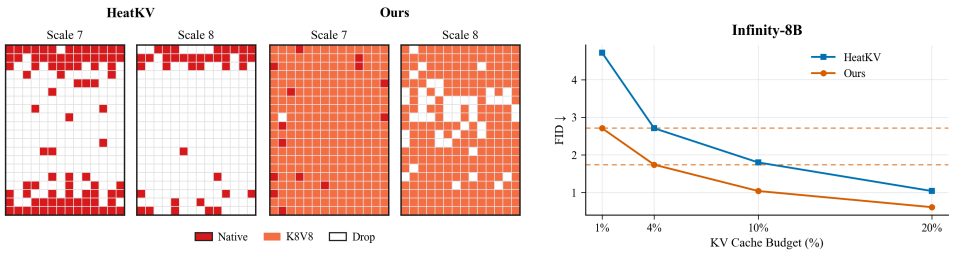
ABSTRACT

Visual autoregressive (VAR) KV caches consume massive GPU memory—over 80 GB for Infinity-8B at 1024×1024 resolution. Existing compression relies on binary pruning: each KV state is either kept at full precision or discarded entirely. Our analysis shows that this forced choice is unnecessary: preserving all tokens at lower precision often proves more effective than discarding a fraction of them, demonstrating that fidelity reduction is not information deletion. This insight points to a different path: rather than deciding only whether to keep a state, we should also decide how faithfully to keep it as its future role evolves. To this end, we propose HorizonKV, which replaces binary decisions with a three-state lifecycle {`Native` (full precision), `K8V8` (8-bit quantized), and `Drop` (released)} governed by the monotonically shrinking remaining generation horizon. Specifically, a small offline calibration first decouples future demand from compression sensitivity for each state. Afterwards, we allocate a global physical-byte budget via Lagrangian relaxation, enforcing irreversible `Native` → `K8V8` → `Drop` transitions as future responsibility decreases. Extensive experiments across Infinity-2B &-8B show that the HorizonKV consistently improves the memory–quality frontier: at only **1%** KV storage, it approaches full-cache visual quality and achieves roughly **4× additional compression** over the best-performing pruning baseline HeatKV (Cederlund et al., 2026) at comparable fidelity, while preserving semantic generation quality on GenEval and DPG-Bench.

1 INTRODUCTION

Visual autoregressive (VAR) models have recently emerged as a powerful paradigm for high-resolution image synthesis, reformulating generation as coarse-to-fine next-scale prediction (Tian

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.



(a) State allocation schematic **(b) FID-budget trade-off**

Figure 2: **State allocation and budget–quality trade-off of our HorizonKV vs. HeatKV (Cederlund et al., 2026).** (a) HeatKV applies binary `Native`/`Drop` decisions to historical KV groups, whereas HorizonKV enables dynamic transitions among `Native`, `K8V8`, and `Drop` based on remaining future demand. (b) FID versus KV cache budget on Infinity-8B (Han et al., 2025). HorizonKV achieves lower FID than HeatKV at all tested budgets.

et al., 2024). In this framework, low-resolution scales first establish global layout and major structures, while subsequent scales progressively refine textures and local details. To reuse information formed at earlier scales, each attention layer repeatedly queries historical key-value (KV) states from preceding scales. As generation proceeds, these historical KV states accumulate rapidly and become a major memory bottleneck—for instance, as illustrated by Figure 1, the full KV cache of Infinity-8B (Han et al., 2025) at 1024×1024 resolution consumes over 80 GB of GPU memory.

Existing approaches to VAR KV cache compression have predominantly focused on pruning (Li et al., 2025a; Xu et al., 2026; Qin et al., 2026b; Cederlund et al., 2026). ScaleKV (Li et al., 2025a) and AMS-KV (Xu et al., 2026) allocate cache budgets across layers and generation scales; HACK (Qin et al., 2026b) and HeatKV (Cederlund et al., 2026) refine selection toward attention heads and head-scale dependencies. Despite increasing granularity in what to retain, all these methods share a fundamental limitation: they treat the retention decision as binary, *i.e.,* **each historical KV state is either kept at full precision or discarded entirely.**

This binary view, we argue, is unnecessarily restrictive. Reducing representation fidelity is functionally distinct from deleting information entirely. As illustrated by Figure 2, our analysis indicates that, given a comparable storage allowance, preserving all tokens at lower precision (`K8V8`, 8-bit quantized) is often more beneficial than keeping a fraction at higher precision (`Native`, full precision), pointing to a qualitative difference between quantization and pruning. In other words, **preserving all historical information at lower precision can be preferable to keeping only a subset at full precision**. Yet existing pruning methods (Xu et al., 2026; Qin et al., 2026b; Cederlund et al., 2026), constrained by the keep-or–evict paradigm, discard KV states that still carry useful visual structure.

To this end, we propose HorizonKV, a horizon-aware KV compression framework that replaces binary keep-or-evict with a three-state lifecycle: `Native`, `K8V8`, and `Drop`. As coarse-to-fine generation proceeds, each historical group's future utility shrinks monotonically—a group used by many remaining scales stays native; one lightly used is quantized; one no longer needed is dropped. To operationalize this, we calibrate each group's future demand and quantization sensitivity offline, then solve a Lagrangian allocation under a global byte budget to produce a frozen schedule of irreversible `Native` → `K8V8` → `Drop` transitions. Experiments on Infinity-2B&-8B models (Han et al., 2025) show that HorizonKV consistently outperforms existing state-of-the-art pruning methods across all budget levels, especially under extreme compression.

Our contributions are summarized as follows:

1) Reframing the problem: We distinguish fidelity reduction from information deletion, arguing that dynamic precision control, rather than binary keep-or-evict, is the key to effective KV compression.

2) Proposed framework: We introduce HorizonKV with a three-state lifecycle (`Native` → `K8V8` → `Drop`), which decouples future demand from compression sensitivity offline and schedules states via Lagrangian optimization under a physical byte budget over the shrinking generation horizon.

3) Superior performance: On Infinity-2B/8B (Han et al., 2025), HorizonKV preserves near-full-cache visual quality at 1% KV budget and achieves $4\times$ additional compression over the strongest

baseline HeatKV (Cederlund et al., 2026) at comparable fidelity, while maintaining semantic generation metrics nearly intact.

2 Related Work

What to Retain: KV Cache Selection and Pruning. Early work on KV cache compression largely framed the problem as a binary selection task: given a fixed budget, decide which tokens or states to keep at full precision and which to evict. In large language models (LLMs), methods such as StreamingLLM (Xiao et al., 2024) and H2O (Zhang et al., 2023) preserve attention sinks or heavy-hitter tokens; FastGen (Ge et al., 2024) and SnapKV (Li et al., 2024) refine selection via profiling or observation-window statistics; PyramidKV (Cai et al., 2025) and Ada-KV (Feng et al., 2025) further adapt budgets across layers or heads.

For VAR models, KV-cache compression has predominantly remained pruning-based, exploiting the coarse-to-fine generation structure. ScaleKV (Li et al., 2025a) and AMS-KV (Xu et al., 2026) allocate budgets across layers and generation scales; HACK (Qin et al., 2026b) and its successor HACK++ (Qin et al., 2026a) incorporate head-aware and historical-scale reliance; HeatKV (Cederlund et al., 2026) ranks head-scale dependencies using offline attention statistics. Despite increasing granularity, all these methods share a fundamental limitation: they treat the retention decision as binary—each KV state is either kept at full precision or discarded entirely. As we show in Section 3.2, this forced choice unnecessarily discards information that remains useful when stored at lower fidelity.

How Precisely to Retain: Quantization and Mixed-Precision. A parallel line of work addresses the fidelity dimension, altering the numerical precision of retained KV states rather than evicting them. KIVI (Liu et al., 2024) studies asymmetric low-bit quantization; KVQuant (Hooper et al., 2024) develops distribution- and outlier-aware quantization schemes; KVTuner (Li et al., 2025b) searches for hardware-friendly mixed-precision configurations. MiKV (Yang et al., 2024) retains would-be-evicted KV at lower precision, while DiffKV (Zhang et al., 2025) unifies selective pruning with mixed-precision quantization. GEAR (Kang et al., 2024) offers a complementary view by combining quantization with low-rank and sparse error correction.

These LLM-oriented methods establish that *whether* to retain and *at what fidelity* are distinct memory-management decisions, but their original designs are not tailored to the scale-wise KV structure of VAR models. This raises a further question: if a historical group's role changes as generation progresses, should its required fidelity also change dynamically over its lifetime?

When and How to Evolve: Adaptive Memory Lifecycles. A third thread, emerging from long-context sequence modeling, treats memory as a dynamic resource whose representation can evolve over time. Transformer-XL (Dai et al., 2019) reuses hidden states via segment-level recurrence; Compressive Transformer (Rae et al., 2020) moves states from short-term to compressed memory; Memorizing Transformers (Wu et al., 2022) store historical KV in external memory with approximate nearest-neighbor retrieval; Infini-attention (Munkhdalai et al., 2024) combines local attention with compressive memory. These methods suggest that historical information need not maintain a fixed representation throughout its entire lifetime.

Closely related to our horizon-aware perspective, LU-KV (Tang et al., 2026) argues that instantaneous importance scores may not reflect long-horizon utility, and instead uses future utility to guide global cache allocation. VAR models provide an unusually explicit setting for this idea: the set of future scales that will query any given historical group is known in advance and shrinks monotonically as coarse-to-fine generation advances. HorizonKV exploits this explicit *shrinking horizon* to dynamically transition each group through `Native` $\rightarrow$ `K8V8` $\rightarrow$ `Drop`, unifying the selection, precision, and temporal evolution axes in a principled, budget-constrained framework.

3 Methodology

3.1 Preliminaries

VAR Modeling. VAR (Tian et al., 2024) formulates image generation as coarse-to-fine next-scale prediction. Given a sequence of scales $\mathcal{R} = (r_0, r_1, \ldots, r_{K-1})$, the generation process factorizes

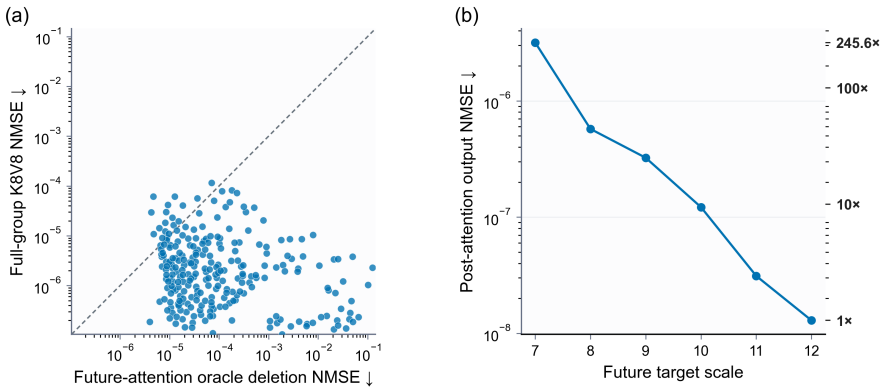


Figure 3: **Controlled intervention analysis on Infinity-2B (Han et al., 2025).** Both panels use the same distortion metric NMSE defined by (3) relative to the Full cache. **(a)** Under matched runtime logical KV budgets, full-group K8V8 quantization is compared with future-attention oracle token pruning, which ranks historical tokens by their aggregated attention from all subsequent target scales. K8V8 yields lower NMSE for ∼**94.0**% groups. **(b)** K8V8-induced NMSE is shown for one historical KV group across future target scales 7–12, with up to a **245.6**× variation.

as:

$$p(\mathcal{R}) = \prod_{t=0}^{K-1} p(r_t \mid r_{<t}), \tag{1}$$

where r_0 denotes the coarsest scale and r_{K-1} the finest. During the generation of target scale r_t, the model attends to historical KV states from all preceding source scales $s < t$. Throughout this paper, we adopt the Infinity family (Han et al., 2025) as our testbed, where keys are stored in FP32 and values in BF16 as the native format.

Group Definition. We partition the KV cache into granular groups at the level of (layer l, head h, source scale s), denoted as $g = (l, h, s)$. Each group corresponds to all KV tokens produced by source scale s within a specific attention head and layer. This grouping strategy is motivated by our empirical observation that both the future utility and compressibility of KV states exhibit systematic variation across these three dimensions, while finer granularities introduce prohibitive scheduling overhead without commensurate quality gains.

State Space. For each group g, we maintain a three-state lifecycle $\mathcal{Q} = \{\texttt{Native}, \texttt{K8V8}, \texttt{Drop}\}$: 1. `Native`: Full-precision storage (FP32 keys + BF16 values), preserving the original representation.

2. `K8V8`: Block-wise INT8 quantization for both keys and values, reducing memory footprint by approximately $4\times$ for keys and $2\times$ for values relative to `Native`, with quantization scales stored as metadata.

3. `Drop`: Complete eviction of the group from the cache.

Different from prior binary keep-or-evict policies, our three-state design allows a group to remain accessible at reduced precision even when full precision is no longer justified by its future utility.

Generation Horizon. At generation stage t, the remaining horizon consists of scales $\{t, t+1, \ldots, K-1\}$ that have not yet been generated. A key observation underlying HorizonKV is that the set of future scales depending on any historical group $g = (l, h, s)$ shrinks monotonically as t advances: only target scales u satisfying $s < t \leq u$ remain relevant. This monotonic structure provides a natural signal for adapting memory fidelity over time.

3.2 Motivating Observations

As illustrated by Figure 3, we start with two empirical observations from experiments that motivate our design.

Fidelity reduction is not equivalent to deletion. With comparable K/V payload, storing complete INT8 groups consistently outperforms retaining only a subset of BF16 tokens across the vast ma-

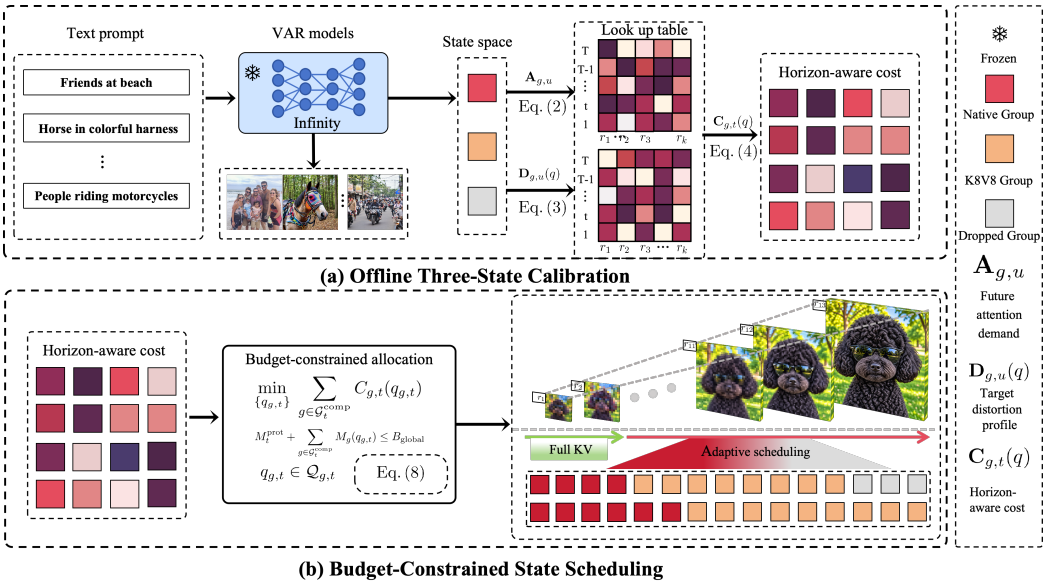


Figure 4: **Overview of HorizonKV.** (a) Offline calibration estimates group–target future demand and state distortion; (b) budget-constrained allocation determines the `Native`/`K8V8`/`Drop` state of each historical group for adaptive scheduling across generation scales.

jority of tested cases, indicating that quantization and pruning operate along fundamentally distinct axes of the memory–quality trade-off.

Required fidelity depends on the remaining target set. Quantization sensitivity varies substantially across future target scales, and groups with similar attention mass can exhibit markedly different distortion profiles.

Taken together, these observations reveal a key insight: no single importance score can adequately answer both *whether* a group will be used and *how faithfully* it should be stored. These two factors must be decoupled and jointly optimized over the shrinking horizon, which we achieve through separate calibration of **future demand** and **compression sensitivity**.

3.3 Offline Three-State Calibration

Figure 4 illustrates the overall pipeline of HorizonKV. The offline calibration phase first characterizes each historical KV group along two independent dimensions: (i) its **future demand** by later target scales, and (ii) its **compression sensitivity**—the degree to which each candidate state perturbs the attention output for those targets. At generation stage t, the calibrated statistics are aggregated exclusively over the remaining target scales $\{t, t+1, \ldots, K-1\}$, yielding a stage-dependent cost that subsequently drives the budgeted state scheduling.

Let $\mathcal{C}$ denote the offline calibration set, with $x \in \mathcal{C}$ a calibration sample. We first estimate the **future demand**. For a valid source–target scale pair $s < u$, let $\mathcal{I}_u$ be the fixed set of query-probe positions at target scale u, and $\mathcal{I}_s$ the key-token indices belonging to source scale s. The native attention mass from target scale u to group $g = (l, h, s)$ is defined as

$$A_{g,u} = \mathbb{E}_{x\sim\mathcal{C}}\Big[\frac{1}{|\mathcal{I}_u|}\sum_{i\in\mathcal{I}_u}\sum_{j\in\mathcal{I}_s}\alpha_{ij}^{(l,h)}(x)\Big], \tag{2}$$

where $\alpha_{ij}^{(l,h)}(x)$ denotes the native attention weight (Li et al., 2025a) from query position i to key position j in layer l and head h, computed with the full-precision cache. Thus, $A_{g,u}$ quantifies the average reliance of future target scale u on the historical group g.

Second, we measure the **compression sensitivity**. We perturb only group g by placing it in a candidate state $q \in \{\texttt{K8V8}, \texttt{Drop}\}$, while keeping all other groups in the `Native` state. Let $\mathbf{z}_{u,i}^{\mathrm{N}}(x)$ be the head response at probe i under the fully native cache, and $\mathbf{z}_{u,i}^{(q;g)}(x)$ the corresponding response when only group g is stored in state q, to capture the functional effect of this perturbation after be-

ing projected into the layer output, we define the distortion metric as the projected attention-output Normalized Mean Square Error (NMSE):

$$D_{g,u}(q) = \mathbb{E}_{x\sim\mathcal{C},i\sim\mathcal{I}_u}\left[\frac{\left\|W_O^{(l,h)}\Delta\mathbf{z}_{u,i}^{(q;g)}(x)\right\|_2^2}{\left\|W_O^{(l,h)}\mathbf{z}_{u,i}^{\mathrm{N}}(x)\right\|_2^2+\epsilon}\right], \qquad D_{g,u}(\texttt{Native}) = 0, \tag{3}$$

where $W_O^{(l,h)}$ is the submatrix of the output projection matrix for layer l and head h, $\Delta\mathbf{z}_{u,i}^{(q;g)}(x) = \mathbf{z}_{u,i}^{\mathrm{N}}(x) - \mathbf{z}_{u,i}^{(q;g)}(x)$ is the perturbation in the head response, and $\epsilon > 0$ is a small constant for numerical stability. This metric measures the relative change in the final attention output, rather than the raw difference in the intermediate head-response space.

Finally, we aggregate the two quantities $\{A_{g,u}, D_{g,u}\}$ over the shrinking generation horizon. At stage t, the **horizon-aware cost** of assigning state q to group g is

$$C_{g,t}(q) = \sum_{u=t}^{K-1} A_{g,u}\cdot\frac{D_{g,u}(q)}{D_{g,u}(\texttt{Drop})+\epsilon}, \tag{4}$$

where $D_{g,u}(\texttt{Drop})$ normalizes the distortion relative to complete eviction, and ϵ prevents division by zero. Note that $C_{g,t}(\texttt{Native}) = 0$. As generation progresses, completed target scales drop out of the summation, causing the same historical group to transition naturally from `Native` to `K8V8` and eventually to `Drop` as its responsibility over the remaining horizon diminishes.

3.4 Budget-Constrained State Scheduling

We designate the KV states from the earliest source scales to be unconditionally preserved in the `Native` format, as they contain few tokens yet encode foundational global structures critical for all subsequent refinements. At generation stage t, the historical groups are partitioned into a protected set $\mathcal{G}_t^{\mathrm{prot}}$ and a compressible set $\mathcal{G}_t^{\mathrm{comp}}$:

$$\mathcal{G}_t^{\mathrm{prot}} = \{g = (l,h,s) \mid 0 \le s < \min(3,t)\}, \qquad \mathcal{G}_t^{\mathrm{comp}} = \{g = (l,h,s) \mid 3 \le s < t\}. \tag{5}$$

Let $M_g(q) = \mathrm{Bytes}(\mathrm{KV}_g^{(q)})$ denote the actual memory footprint of state q, including all quantization metadata, with $M_g(\texttt{Drop}) = 0$. The protected footprint is then

$$M_t^{\mathrm{prot}} = \sum_{g\in\mathcal{G}_t^{\mathrm{prot}}} M_g(\texttt{Native}), \tag{6}$$

and the global physical budget is defined as

$$B_{\mathrm{global}} = \rho\cdot M_{\mathrm{peak,Full}}, \tag{7}$$

where ρ is the target compression ratio and $M_{\mathrm{peak,Full}}$ is the peak memory of the uncompressed full KV cache. This budget is enforced uniformly across all stages, ensuring that the optimization is subject to a consistent physical memory cap rather than stage-specific proxies.

Critically, state transitions are **irreversible**: $\texttt{Native} \succ \texttt{K8V8} \succ \texttt{Drop}$. Once a group is quantized or evicted, the discarded higher-precision information cannot be recovered. We therefore define $\mathcal{Q}_{g,t} \subseteq \{\texttt{Native}, \texttt{K8V8}, \texttt{Drop}\}$ as the set of states still reachable for group g at stage t.

The scheduling problem is now to select, for each group, a reachable state that maximizes the preserved utility over the remaining horizon while respecting the byte budget. Using the cost in (4), we solve the following constrained optimization:

$$\min_{\{q_{g,t}\}} \textstyle\sum_{g\in\mathcal{G}_t^{\mathrm{comp}}} C_{g,t}(q_{g,t}), \text{s.t. } M_t^{\mathrm{prot}} + \sum_{g\in\mathcal{G}_t^{\mathrm{comp}}} M_g(q_{g,t}) \le B_{\mathrm{global}}, \quad q_{g,t}\in\mathcal{Q}_{g,t}. \tag{8}$$

This formulation cleanly decouples the quality objective (minimizing future distortion) from the physical memory constraint, enabling a principled comparison across different state configurations.

To make this problem tractable, we apply Lagrangian relaxation (Bertsekas, 2015). We introduce a non-negative multiplier $\lambda \ge 0$ for the budget constraint and form the Lagrangian:

$$\mathcal{L}(\{q_{g,t}\},\lambda) = \sum_{g\in\mathcal{G}_t^{\mathrm{comp}}} C_{g,t}(q_{g,t}) + \lambda\Big(\sum_{g\in\mathcal{G}_t^{\mathrm{comp}}} M_g(q_{g,t}) - (B_{\mathrm{global}} - M_t^{\mathrm{prot}})\Big). \tag{9}$$

Table 1: Quantitative comparisons of output consistency on MS-COCO 2017.

		Infinity-2B				Infinity-8B			
Method	Budget	KV Cache	FID↓	LPIPS↓	PSNR↑	KV Cache	FID↓	LPIPS↓	PSNR↑
Full Cache	100%	38550 MB	–	–	–	84328 MB	–	–	–
StreamingLLM	20%	7800 MB	3.52	0.139	22.00	17062 MB	3.29	0.111	22.32
HACK	20%	7800 MB	3.29	0.122	22.79	17062 MB	2.80	0.110	21.99
SnapKV	20%	7800 MB	3.11	0.108	23.26	17062 MB	3.09	0.102	22.72
ScaleKV	20%	7800 MB	1.94	0.079	24.93	17062 MB	1.44	0.059	25.60
HeatKV	20%	7800 MB	1.02	0.044	28.43	17062 MB	1.04	0.043	27.44
HorizonKV	20%	7800 MB	**0.92**	**0.041**	**29.35**	17062 MB	**0.61**	**0.027**	**30.48**
StreamingLLM	10%	3900 MB	4.87	0.199	19.58	8531 MB	4.78	0.152	20.57
HACK	10%	3900 MB	4.30	0.162	21.05	8531 MB	3.80	0.135	20.94
SnapKV	10%	3900 MB	4.42	0.148	21.56	8531 MB	4.77	0.144	20.65
ScaleKV	10%	3900 MB	2.59	0.113	22.88	8531 MB	2.13	0.085	23.25
HeatKV	10%	3900 MB	1.67	0.073	25.40	8531 MB	1.80	0.072	24.41
HorizonKV	10%	3900 MB	**1.23**	**0.053**	**27.31**	8531 MB	**1.04**	**0.045**	**27.37**
StreamingLLM	4%	1590 MB	6.15	0.239	18.55	3478 MB	6.57	0.197	19.11
HACK	4%	1590 MB	5.82	0.231	18.73	3478 MB	5.57	0.185	19.34
SnapKV	4%	1590 MB	4.85	0.183	20.18	3478 MB	6.57	0.185	19.21
ScaleKV	4%	1590 MB	4.92	0.168	20.88	3478 MB	3.71	0.137	20.69
HeatKV	4%	1590 MB	2.70	0.127	22.14	3478 MB	2.71	0.111	21.78
HorizonKV	4%	1590 MB	**2.23**	**0.104**	**23.29**	3478 MB	**1.74**	**0.075**	**23.99**
HeatKV	1%	385.5 MB	6.39	0.246	18.31	843.3 MB	4.72	0.203	18.38
HorizonKV	1%	385.5 MB	**3.36**	**0.163**	**20.62**	843.3 MB	**2.71**	**0.118**	**21.31**

Since the term $-\lambda(B_{\text{global}} - M_t^{\text{prot}})$ is constant with respect to the state variables, minimizing the Lagrangian decouples across groups:

$$q_{g,t}^*(\lambda) = \arg\min_{q \in \mathcal{Q}_{g,t}} \{C_{g,t}(q) + \lambda M_g(q)\}. \tag{10}$$

4 Experimental Analysis

4.1 Experimental Setup

Base Models. We evaluate HorizonKV on two VAR-based text-to-image models with different capacities, Infinity-2B and Infinity-8B (Han et al., 2025), to validate its effectiveness across model scales. We consider four physical KV-cache budgets: 20%, 10%, 4%, and 1% of the original cache size, covering both moderate and highly aggressive compression regimes. All budgets include the encoded KV payload and the associated quantization metadata.

Dataset and Metrics. We evaluate reconstruction consistency with Full Cache on the MS-COCO 2017 validation set (Lin et al., 2014). All methods use identical prompts, random seeds, and generation settings to ensure matched comparisons. Reconstruction fidelity is measured using FID (Heusel et al., 2017), LPIPS (Zhang et al., 2018), and PSNR against the corresponding Full-Cache generations. Besides, we evaluate GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024), which cover object composition, attributes, spatial relations, and prompt alignment. System-level experiments are conducted on an NVIDIA RTX PRO 6000. We use 10 fixed calibration samples that are disjoint from the evaluation data.

Compression Baselines. We compare HorizonKV with representative KV-cache compression methods, including StreamingLLM (Xiao et al., 2024), SnapKV (Li et al., 2024), HACK (Qin et al., 2026b), ScaleKV (Li et al., 2025a), and HeatKV (Cederlund et al., 2026). These methods cover static retention, attention-based token selection, head-aware compression, scale-aware allocation, and fine-grained head–scale cache selection. All reconstruction comparisons within the same budget use strictly matched physical KV-cache budgets.

4.2 Main Results

Comparison with Compression Baselines. Table 1 compares reconstruction fidelity under strictly matched physical KV-cache budgets. HorizonKV consistently outperforms existing meth-

Table 2: Quantitative comparisons on GenEval and DPG-Bench.

Methods	KV Budget	GenEval				DPG		
		Two Obj.	Position	Color Attr.	Overall↑	Global	Relation	Overall↑
Infinity-2B	100%	0.77	0.26	0.54	0.670	84.50	92.13	82.41
+ HeatKV	4%	0.78	**0.27**	0.54	0.673	84.19	92.21	**82.34**
+ HorizonKV	4%	0.78	0.26	**0.55**	**0.675**	84.19	**92.40**	82.26
+ HeatKV	1%	0.78	**0.27**	**0.54**	0.668	**84.80**	**92.98**	81.55
+ HorizonKV	1%	**0.79**	0.26	0.53	0.668	83.28	92.40	**82.30**
Infinity-8B	100%	0.95	0.58	0.76	0.820	86.02	94.41	86.27
+ HeatKV	4%	0.95	0.57	0.74	0.820	85.71	94.45	86.15
+ HorizonKV	4%	**0.96**	0.57	**0.75**	**0.821**	**86.02**	**94.56**	**86.22**
+ HeatKV	1%	**0.97**	**0.58**	0.73	0.822	85.71	94.37	85.96
+ HorizonKV	1%	0.96	0.57	**0.75**	0.822	**86.02**	**94.60**	**86.08**

ods on both Infinity-2B and Infinity-8B across all evaluated budgets, achieving the **lowest** FID and LPIPS together with the **highest** PSNR.

The advantage of HorizonKV becomes increasingly pronounced as the cache budget tightens. At a 4% budget, HorizonKV outperforms the strongest baseline, HeatKV, reducing the FID from 2.70 to 2.23 on Infinity-2B and from 2.71 to 1.74 on Infinity-8B, corresponding to **absolute improvements of 0.47 and 0.97**, respectively. Under the more aggressive 1% budget, the FID is further reduced from 6.39 to 3.36 on Infinity-2B and from 4.72 to 2.71 on Infinity-8B, yielding **absolute improvements of 3.03 and 2.01**, respectively. These results show that HorizonKV's advantage becomes more pronounced under tighter cache budgets.

Notably, on Infinity-8B, HorizonKV@1% and HeatKV@4% achieve the same FID of 2.71. HorizonKV therefore reaches comparable reconstruction fidelity with **only 25% of the KV-cache budget**. This suggests that, under tight memory constraints, controlling the representation fidelity of retained history provides a more favorable memory–fidelity trade-off than deciding retention alone.

Results on GenEval and DPG. We further evaluate whether the reconstruction gains translate to semantic generation quality using GenEval and DPG-Bench. As shown in Table 2, HorizonKV largely preserves the semantic capability of the original model even under aggressive KV-cache compression. On Infinity-8B, HorizonKV@4% achieves a GenEval Overall score of 0.821, slightly higher than the 0.820 of Full Cache, while DPG decreases only marginally from 86.27 to 86.22. At a 1% budget, HorizonKV reaches 0.822 on GenEval Overall while maintaining a DPG score of 86.08, higher than the 85.96 of HeatKV@1%. A similar trend is observed on Infinity-2B: HorizonKV@4% achieves a GenEval Overall score of 0.675, higher than both Full Cache (0.670) and HeatKV@4% (0.673). At a 1% budget, HorizonKV reaches 82.30 on DPG, 0.75 points higher than HeatKV (81.55), while maintaining a comparable GenEval overall score of 0.668.

Qualitative Results. Figure 5 compares the intermediate generations of Full Cache, HeatKV@1%, and HorizonKV@1% across multiple scales. At the early coarse scales (12×12 and 16×16), all methods exhibit similar global structures. Starting from Scale 9 (32×32), however, HeatKV develops visible distortions in the hand region, particularly in the finger structure, which persist into the final scale. In contrast, HorizonKV preserves a more coherent local structure and remains closer to Full Cache throughout the refinement process.

4.3 Analytical Experiments

Impact of Memory State Space. As shown in Figure 6 (a), `Native/Drop` obtains an FID of 2.55. Introducing `K4V4` reduces FID to 2.32, while `Native/K8V8/Drop` achieves the **best** result of 2.23. This validates that expanding the state space further does not yield monotonic gains. The tested four-state and five-state variants obtain FID scores of 2.37 and 2.30, respectively, both falling short of `Native/K8V8/Drop`. Intermediate fidelity states therefore improve the binary `Native/Drop` formulation, but a richer state space also makes global allocation more difficult.

Component Ablation. Figure 6 (b) further isolates the contributions of horizon awareness and three-state fidelity allocation. Starting from Static Binary, introducing only horizon awareness reduces FID from 4.63 to 2.55, while introducing only the three-state representation reduces it to 3.18. Combining both components yields the full HorizonKV, which achieves the **best** FID of 2.23. These



Figure 5: **Qualitative comparison of Full Cache, HeatKV@1%, and HorizonKV@1% across generation scales.** HeatKV begins to exhibit visible finger distortion around Scale 9 (32×32), whereas HorizonKV better preserves the local hand structure through the final refinement stage.

results show that horizon awareness and three-state fidelity allocation provide complementary gains: the former dynamically adjusts the remaining future demand over the shrinking generation horizon, while the latter allows useful historical KV states to remain available at reduced fidelity before being eventually released.

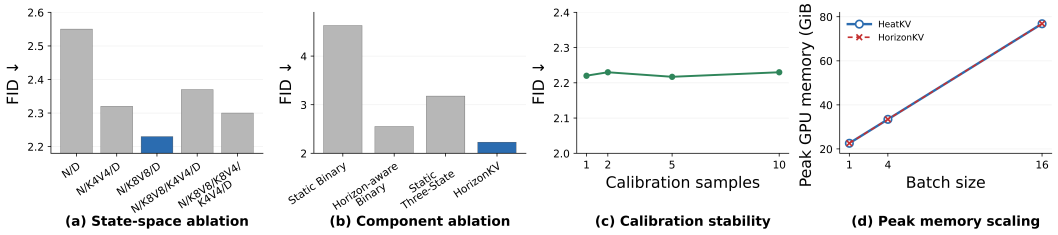


Figure 6: **Analytical and system-level experiments at a 4% KV-cache budget.** (a) Impact of the memory state space. (b) Component ablation of horizon awareness and three-state fidelity allocation. (c) Calibration stability with different numbers of calibration samples. (d) Peak GPU memory across batch sizes.

Calibration Stability. Figure 6 (c) studies the sensitivity of HorizonKV to the calibration-set size. The FID remains highly stable as the number of calibration samples increases from 1 to 10: 2.220, 2.230, 2.217, and 2.230 for 1, 2, 5, and 10 samples, respectively. The overall variation is only about 0.013 FID, showing that HorizonKV requires only a very small calibration set to obtain a stable compression schedule.

Memory Efficiency Analysis. As shown in Figure 6 (d), at a 4% KV-cache budget, HeatKV/HorizonKV reach peak GPU memory usages of 22.57/22.57, 33.43/33.43, and 76.86/76.85 GiB for batch sizes 1, 4, and 16, respectively. The nearly overlapping memory traces indicate that HorizonKV does not rely on hidden Native-KV copies, and that its compressed-state schedule corresponds to real reductions in persistent GPU storage.

5 Conclusion

In this paper, we introduce HorizonKV, a horizon-aware framework that replaces binary KV retention with a three-state lifecycle (`Native`, `K8V8`, `Drop`) governed by the shrinking generation horizon. By decoupling future demand from compression sensitivity and enforcing irreversible transitions under a physical byte budget, HorizonKV consistently outperforms a variety of state-of-the-art pruning methods, achieving comparable fidelity with 4× less memory than the best-performing HeatKV (Cederlund et al., 2026) at extreme compression. Our results highlight that effective KV cache management must control both which information survives and how faithfully it survives.

AI USE STATEMENT

Generative AI tools were used solely for language polishing and improving the clarity and readability of the manuscript. All scientific content, experiments, results, and conclusions were independently developed and verified by the authors.

REPRODUCIBILITY STATEMENT

We aim to facilitate reproducibility as follows. The proposed method and implementation procedure are described in detail in the main text. The model configurations, datasets, compression budgets, evaluation metrics, calibration protocol, and hardware settings are provided in the experimental setup, while additional implementation details and ablation settings are given in the Appendix. Calibration and evaluation samples are kept disjoint, and all comparative experiments use matched prompts, random seeds, and generation configurations. To further support reproducibility, we provide the full source code, configuration files, and reproduction instructions in our anonymous repository: `https://anonymous.4open.science/r/HorizonKV-DBBB`.

REFERENCES

Dimitri P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025.

Jonathan Cederlund, Axel Berg, William Isaksson, Durmus Alp Emre Acar, Chuteng Zhou, and Pontus Giselsson. HeatKV: Head-tuned KV-cache compression for visual autoregressive modeling. *arXiv preprint arXiv:2605.14877*, 2026.

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

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15733–15744, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip diffusion models with LLM for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In *Proceedings of the 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 305–321. PMLR, 2024.

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware KV cache compression. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36451–36485. PMLR, 2025b.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with Infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Ziran Qin, Yuchen Jiang, Mingbao Lin, Youru Lv, Hang Guo, Wen Fei, and Weiyao Lin. HACK++: Towards more effective head-aware key-value compression for efficient visual autoregressive modeling. *arXiv preprint arXiv:2606.08302*, 2026a.

Ziran Qin, Youru Lv, Mingbao Lin, Hang Guo, Zeren Zhang, Danping Zou, and Weiyao Lin. Head-aware KV cache compression for efficient visual autoregressive modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24982–24990, 2026b.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Ziyao Tang, Pengkun Jiao, Xinhang Chen, Wei Liu, Shiyong Li, and Jingjing Chen. Predicting future utility: Global combinatorial optimization for task-agnostic KV cache eviction. In *Forty-third International Conference on Machine Learning*, 2026.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022.

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

Boxun Xu, Yu Wang, Zihu Wang, and Peng Li. AMS-KV: Adaptive KV caching in multi-scale visual autoregressive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27206–27214, 2026.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C. S. Lui, and Haibo Chen. DiffKV: Differentiated memory management for large language models with parallel KV compaction. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, pp. 431–445, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A Implementation and Calibration Details

A.1 KV Representation, Quantization, and Cache Organization

Physical KV representation. Infinity stores native keys in FP32 and values in BF16. For a historical KV group $g = (l, h, s)$, with T_s source tokens and head dimension d_h, the persistent Native representation requires

$$M_g(\texttt{Native}) = 48 T_s d_h \text{ bits.} \tag{A1}$$

K8V8 state. In `K8V8`, both keys and values are stored as signed INT8 payloads together with FP16 quantization scales. Quantization is performed independently for keys and values, for each attention head, and for each effective batch element. Along the token dimension, every consecutive $b = 8$ tokens form one quantization block. Let $X \in \mathbb{R}^{T_s \times d_h}$ denote the key or value tensor of one attention head. For block X_r, the quantization scale is

$$s_r = \mathrm{FP16}\left[\max\left(\frac{\max|X_r|}{127}, 2^{-24}\right)\right], \tag{A2}$$

and the corresponding INT8 payload is

$$Q_r = \mathrm{INT8}\left[\mathrm{clip}\left(\mathrm{round}\left(\frac{X_r}{\mathrm{FP32}(s_r)}\right), -127, 127\right)\right]. \tag{A3}$$

We use symmetric quantization and therefore do not maintain an explicit zero point. Keys and values use the same codec but keep independent scales. For the Infinity configuration used in our experiments, $d_h = 128$, so one quantization scale typically covers 8×128 values. During attention computation, a compressed block is reconstructed as $\hat{X}_r = s_r Q_r$ and converted to the numerical format required by the attention computation. The reconstructed floating-point tensor is temporary. Once a group transitions to `K8V8`, the persistent cache retains only the compressed payload and its scales rather than an additional `Native` copy. For `Drop`, the historical entry is removed from the persistent cache and excluded from subsequent historical attention. Thus, `Drop` corresponds to actual cache eviction rather than retaining a dense tensor and merely masking its contribution.

State-dependent storage. Let $n_s^{\mathrm{blk}} = \lceil T_s/8 \rceil$. The persistent storage of one `K8V8` group is

$$M_g(\texttt{K8V8}) = 16 T_s d_h + 32 n_s^{\mathrm{blk}} \text{ bits,} \tag{A4}$$

where the first term corresponds to INT8 keys and values and the second term accounts for the FP16 key and value scales. The `Drop` state has

$$M_g(\texttt{Drop}) = 0. \tag{A5}$$

Including an effective attention-batch dimension B_{eff}, the same costs can equivalently be written as

$$M_g(\texttt{Native}) = 6 B_{\mathrm{eff}} T_s d_h \text{ bytes,} \tag{A6}$$

$$M_g(\texttt{K8V8}) = B_{\mathrm{eff}}\left(2 T_s d_h + 4 n_s^{\mathrm{blk}}\right) \text{ bytes.} \tag{A7}$$

All reported KV-cache budgets include both the encoded K/V payload and quantization scales. Temporary reconstruction tensors, attention workspaces, and allocator overhead are not included in the persistent per-state cache cost.

Cache organization. The scheduling unit $g = (l, h, s)$ maps directly to the runtime cache, where l, h, s identify the attention layer, head, and historical source scale. Each group therefore contains the complete historical K/V representation produced by one source scale for one attention head. The earliest three source scales, $s \in \{0, 1, 2\}$, form the protected prefix. Under the scale configuration used in our experiments, they contain only $1 + 4 + 16 = 21$ tokens per head. Their memory contribution is small compared with later high-resolution scales, while they provide coarse structural context to many subsequent targets. These groups therefore remain `Native` throughout their useful lifetime, whereas later source scales are managed by the frozen schedule.

A.2 Calibration of Future Demand and Compression Sensitivity

Calibration converts a small set of `Native` generation trajectories into the two quantities used by the offline scheduler: Future Demand $A_{g,u}$ (2) and Compression Sensitivity $D_{g,u}(q)$ (3). We use

10 fixed prompt/seed pairs disjoint from evaluation, and no model parameters are updated during calibration.

Future-demand collection: For a valid source-target pair $s < u$, Native attention probabilities are collected from the corresponding layer and head. Let $\mathcal{I}_s$ denote the key indices belonging to historical source scale s, and let $\mathcal{I}_u$ denote the target queries associated with target scale u. The precise definition follows (2) in the main paper.

A.3 Offline-to-Runtime Execution Pipeline

HorizonKV separates calibration and allocation from runtime execution: Native calibration $\rightarrow$ (A, D) $\rightarrow$ offline schedule construction $\rightarrow$ frozen schedule $\rightarrow$ runtime execution. Future Demand and Compression Sensitivity are collected once. The offline scheduler constructs a target-stage-specific assignment S_t for each generation stage t. The resulting schedule is frozen before evaluation, and all evaluation prompts share the same schedule. No prompt-specific importance estimation or online allocation is performed during generation. At runtime, stage t reads S_t. Retained `K8V8` groups are reconstructed when accessed by attention, dropped groups are omitted, and newly generated KV initially remains in the model's Native representation. The runtime sequence is $S_t \rightarrow$ decode retained history $\rightarrow$ append current KV $\rightarrow$ attention $\rightarrow$ transition toward S_{t+1}.

B Budget-Constrained Schedule Construction

B.1 Stage-wise Costs and Physical Budget

At target stage t, only historical groups satisfying $s < t$ exist, while only unfinished targets $u \in \{t, t+1, \ldots, K-1\}$ remain relevant. For every active compressible group g and candidate state q,

$$C_{g,t}(q) = \sum_{u=t}^{K-1} A_{g,u} \frac{D_{g,u}(q)}{D_{g,u}(\texttt{Drop}) + \epsilon}. \tag{A8}$$

The global physical cache budget is

$$B_{\text{global}} = \rho M_{\text{peak,Full}}. \tag{A9}$$

At stage t, the budget available to compressible groups is

$$B_t^{\text{alloc}} = B_{\text{global}} - \sum_{g \in \mathcal{G}_t^{\text{prot}}} M_g(\texttt{Native}). \tag{A10}$$

The same physical cap B_{global} is used across all target stages.

B.2 Lagrangian Allocation and Residual Refinement

For multiplier $\lambda \geq 0$, each compressible group independently selects

$$q_{g,t}^*(\lambda) = \arg\min_{q \in \mathcal{Q}_{g,t}} \left\{ C_{g,t}(q) + \lambda M_g(q) \right\}. \tag{A11}$$

The corresponding cache usage is

$$M_t(\lambda) = \sum_{g \in \mathcal{G}_t^{\text{comp}}} M_g(q_{g,t}^*(\lambda)). \tag{A12}$$

We initialize $\lambda_{\text{low}} = 0, \lambda_{\text{high}} = 1$, increase λ_{high} until the allocation is feasible, and then use 96 binary-search iterations to obtain the final feasible multiplier. When multiple states yield the same augmented cost, the lower-memory state is preferred:

$$\texttt{Native} \succ \texttt{K8V8} \succ \texttt{Drop}. \tag{A13}$$

For a feasible adjacent promotion $q_{g,t} \rightarrow q_{g,t}^+$, we define

$$\Delta C_g = C_{g,t}(q_{g,t}) - C_{g,t}(q_{g,t}^+), \qquad \Delta M_g = M_g(q_{g,t}^+) - M_g(q_{g,t}). \tag{A14}$$

Residual capacity is filled by feasible promotions ranked according to $\frac{\Delta C_g}{\Delta M_g}$.

B.3 Irreversible Schedule Construction

The state lifecycle is irreversible:

$$\texttt{Native} \rightarrow \texttt{K8V8} \rightarrow \texttt{Drop}, \tag{A15}$$

while a `Native` group may also transition directly to `Drop`. Accordingly,

$$\mathcal{Q}_{g,t} = \begin{cases} \{\texttt{Native}, \texttt{K8V8}, \texttt{Drop}\}, & q_{g,t-1} = \texttt{Native}, \\ \{\texttt{K8V8}, \texttt{Drop}\}, & q_{g,t-1} = \texttt{K8V8}, \\ \{\texttt{Drop}\}, & q_{g,t-1} = \texttt{Drop}. \end{cases} \tag{A16}$$

The detailed algorithm is summarized in Algorithm B.3.

Algorithm B.3 Horizon-Aware Schedule Construction

Require: $A_{g,u}$, $D_{g,u}(q)$, $M_g(q)$, B_{global}
Ensure: $\mathcal{S} = \{S_t\}_{t=1}^{K-1}$

$\mathcal{S} \leftarrow \emptyset$
for $t = 1, \ldots, K-1$ **do**
  Construct $\mathcal{G}_t^{\text{prot}}$, $\mathcal{G}_t^{\text{comp}}$, and B_t^{alloc}
  for $g \in \mathcal{G}_t^{\text{comp}}$ **do**
    Construct reachable state set $\mathcal{Q}_{g,t}$ from $q_{g,t-1}$
    for $q \in \mathcal{Q}_{g,t}$ **do**
      $C_{g,t}(q) \leftarrow \sum_{u=t}^{K-1} A_{g,u} \dfrac{D_{g,u}(q)}{D_{g,u}(\texttt{Drop}) + \epsilon}$
    end for
  end for
  Find budget-feasible λ_t by binary search
  for $g \in \mathcal{G}_t^{\text{comp}}$ **do**
    $q_{g,t} \leftarrow \arg\min_{q \in \mathcal{Q}_{g,t}} \{C_{g,t}(q) + \lambda_t M_g(q)\}$
  end for
  $R_t \leftarrow B_t^{\text{alloc}} - \sum_{g \in \mathcal{G}_t^{\text{comp}}} M_g(q_{g,t})$
  while a feasible adjacent promotion fits in R_t **do**
    $g^\star \leftarrow \arg\max_g \dfrac{\Delta C_g}{\Delta M_g}$
    Apply $q_{g^\star,t} \rightarrow q^+_{g^\star,t}$ and update R_t
  end while
  Set protected groups to `Native` and store assignment S_t
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_t\}$
end for
return $\mathcal{S}$

C Relation to KV Quantization and Mixed-Precision Compression

C.1 Relation to Existing KV Quantization

A parallel line of KV-cache compression focuses on the numerical fidelity of retained states. KIVI (Liu et al., 2024), KVQuant (Hooper et al., 2024), and KVTuner (Li et al., 2025b) study low-bit or mixed-precision KV representations, while MiKV (Yang et al., 2024) and DiffKV (Zhang et al., 2025) combine reduced precision with selective retention. GEAR (Kang et al., 2024) further explores quantization-error correction through additional low-rank or sparse components. These methods establish an important principle shared with HorizonKV: deciding whether historical information remains available and deciding how faithfully it is represented are distinct memory-management

dimensions. Their original formulations are primarily developed for autoregressive language-model KV caches. Visual autoregressive generation exposes an additional source-target structure:

$$\text{historical source scale} \rightarrow \text{future target scales}.$$

HorizonKV focuses on this scheduling dimension. Its contribution is not a new low-bit codec itself, but a mechanism for determining when a historical group should remain Native, when reduced fidelity is sufficient, and when it can be removed.

C.2 Why Pure Quantization Is Insufficient at Extreme Budgets

Infinity uses FP32 keys and BF16 values, corresponding to

$$32 + 16 = 48 \text{ bits per key/value element pair before compression.}$$

If the complete historical cache were retained uniformly, the ideal raw-payload ratios would be

$$33.33\% \text{ for } \texttt{K8V8}, \quad 16.67\% \text{ for } \texttt{K4V4}, \quad 8.33\% \text{ for } \texttt{K2V2}, \quad 4.17\% \text{ for } \texttt{K1V1},$$

already ignoring quantization scales and additional metadata. Therefore, a strict 4% budget, and particularly a 1% budget, cannot generally be achieved by preserving the complete historical cache and uniformly reducing numerical precision. Extreme compression must jointly determine which historical information remains and at what fidelity it remains.

D Additional Experiments and Detailed Analysis

D.1 Detailed Motivation Experiments

Experimental setup. We conduct the controlled intervention analysis on Infinity-2B using 10 fixed prompt/seed pairs. We evaluate 384 historical KV groups across 8 layers $\times$ 8 heads $\times$ 6 source scales. For every intervention, all groups except the tested $g = (l, h, s)$ remain Native. Full-Cache inference provides the reference trajectory and future-attention statistics, after which each intervention is evaluated relative to the same Native trajectory.

Full-group K8V8 versus future-attention oracle deletion. For oracle deletion, each historical token j is ranked using its aggregated attention over all future target scales:

$$I_j^{(g)} = \sum_{u>s} \sum_{i \in \mathcal{I}_u} \alpha_{ij}^{\text{Full}}. \tag{A17}$$

Across the 384 tested groups, `K8V8` produces lower projected-output distortion than future-attention oracle deletion for $361/384 = 94.0\%$ of the groups. The median NMSE is 1.68×10^{-6} for full-group K8V8 and 3.56×10^{-5} for oracle deletion. Their observed ranges are $1.08 \times 10^{-8} - 1.16 \times 10^{-4}$ and $3.98 \times 10^{-6} - 2.94$, respectively. The median group-wise error ratio is approximately 0.043, corresponding to about $23.4\times$ lower distortion for `K8V8`.

Future-conditioned fidelity. For $g = (l = 8, h = 11, s = 6)$, the projected-output NMSE changes from 3.17×10^{-6} at target scale 7 to

$$5.73 \times 10^{-7},\ 3.24 \times 10^{-7},\ 1.22 \times 10^{-7},\ 3.12 \times 10^{-8},\ 1.29 \times 10^{-8}$$

at target scales 8–12, respectively. The maximum variation is approximately $245.6\times$. Thus, the fidelity required by a historical representation can vary substantially with the future target that consumes it.

D.2 Detailed GenEval Results

The complete GenEval category breakdown is reported in Table 3. Scores are reported on the same normalized 0–1 scale used in the main paper. The Two Object, Position, Color Attribution, and Overall entries follow exactly the displayed values in the main semantic comparison. Bold values indicate the better displayed result between HeatKV and HorizonKV under the same model size and KV budget; equal displayed values are left unbolded.

On Infinity-2B, Full obtains an Overall GenEval score of 0.670. HorizonKV retains 0.675 at 4% and 0.668 at 1% compared with 0.673 and 0.668 for HeatKV at the corresponding budgets. On

Table 3: Detailed GenEval results on Infinity-2B and Infinity-8B.

Methods	KV Budget	Single Obj.	Two Obj.	Counting	Colors	Position	Color Attr.	Overall↑
Infinity-2B	100%	1.00	0.77	0.63	0.82	0.26	0.54	0.670
+ HeatKV	4%	0.99	0.78	0.63	0.83	**0.27**	0.54	0.673
+ HorizonKV	4%	**1.00**	0.78	**0.64**	0.83	0.26	**0.55**	**0.675**
+ HeatKV	1%	0.98	0.78	0.63	0.81	**0.27**	**0.54**	0.668
+ HorizonKV	1%	**0.99**	**0.79**	0.63	**0.82**	0.26	0.53	0.668
Infinity-8B	100%	1.00	0.95	0.76	0.88	0.58	0.76	0.820
+ HeatKV	4%	1.00	0.95	0.78	0.87	0.57	0.74	0.820
+ HorizonKV	4%	1.00	**0.96**	0.78	0.87	0.57	**0.75**	**0.821**
+ HeatKV	1%	1.00	**0.97**	0.78	0.88	**0.58**	0.73	0.822
+ HorizonKV	1%	1.00	0.96	0.78	0.88	0.57	**0.75**	0.822

Table 4: Detailed DPG-Bench results on Infinity-2B and Infinity-8B.

Methods	KV Budget	Global	Relation	Entity	Attribute	Other	Overall↑
Infinity-2B	100%	84.50	92.13	88.60	86.06	84.00	82.41
+ HeatKV	4%	84.19	92.21	**88.69**	85.92	82.40	**82.34**
+ HorizonKV	4%	84.19	**92.40**	88.40	**86.08**	82.40	82.26
+ HeatKV	1%	**84.80**	**92.98**	88.19	**85.90**	82.00	81.55
+ HorizonKV	1%	83.28	92.40	**88.39**	85.74	**82.40**	**82.30**
Infinity-8B	100%	86.02	94.41	91.24	89.14	83.60	86.27
+ HeatKV	4%	85.71	94.45	91.08	89.10	**86.00**	86.15
+ HorizonKV	4%	**86.02**	**94.56**	**91.40**	**89.24**	82.80	**86.22**
+ HeatKV	1%	85.71	94.37	90.98	89.02	**84.80**	85.96
+ HorizonKV	1%	**86.02**	**94.60**	**91.27**	**89.16**	84.00	**86.08**

Infinity-8B, Full obtains 0.820 Overall. HorizonKV retains 0.821 at 4% and 0.822 at 1% compared with 0.820 and 0.822 for HeatKV, respectively. Overall, semantic alignment remains stable even at the most aggressive cache budgets.

D.3 Detailed DPG-Bench Results

The DPG-Bench evaluation provides both high-level semantic dimensions and the complete fine-grained category breakdown. Bold values indicate the better result between HeatKV and HorizonKV under the same model size and KV budget. Equal values are left unbolded.

On Infinity-2B, HorizonKV retains Overall DPG scores of 82.26 at 4% and 82.30 at 1%, compared with 82.41 for Full. At 1%, HorizonKV improves Overall from 81.55 for HeatKV to 82.30. On Infinity-8B, HorizonKV $@4\%$ retains an Overall score of 86.22 compared with 86.27 for Full and 86.15 for HeatKV. At 1%, HorizonKV retains 86.08 compared with 85.96 for HeatKV. At 4%, HorizonKV also improves Relation, Entity, Attribute, Non-spatial, and Spatial over HeatKV, while HeatKV is higher on the Other category. Overall, the semantic evaluations are consistent with stable prompt alignment under aggressive KV-cache compression.

E Memory Accounting and Runtime Details

E.1 Physical KV Budget, Peak Memory, and Latency

The reported 20%, 10%, 4%, and 1% budgets refer specifically to persistent historical KV storage relative to the Full-Cache historical peak (A9): The accounting includes state-dependent K/V payload and quantization scales. A `K8V8` group is therefore charged for both its INT8 payload and FP16 scales, while a dropped group has zero persistent cache cost. This historical KV budget should not be confused with end-to-end GPU memory. Peak GPU memory additionally contains model parameters, activations, current-scale K/V tensors, attention intermediates, temporary reconstructed historical states, state-transition workspaces, and allocator overhead. At the 4% cache budget, HeatKV and HorizonKV use 22.57/22.57 GiB at batch size 1, 33.43/33.43 GiB at batch size 4, and 76.86/76.85 GiB at batch size 16, respectively.

Latency. We additionally measure end-to-end generation latency on Infinity-8B using a single RTX PRO 6000. At batch size 1, Full-Cache generation takes 2.101 s/image, while HeatKV $@4\%$ and HorizonKV $@4\%$ require 2.199 and 2.869 s/image, respectively. At batch size 4, HeatKV

Table 5: Fine-grained DPG-Bench category results.

Methods	KV	Color	Attr.-other	Shape	Size	Texture	Ent.-part	Ent.-state	Ent.-whole	Count	Text	Non-sp.	Spatial
Infinity-2B	100%	89.26	82.39	75.11	73.55	87.54	83.98	81.84	90.45	84.00	84.00	88.68	92.36
+ HeatKV	4%	88.96	**82.95**	**76.42**	73.14	87.05	83.40	**82.79**	**90.45**	82.00	84.00	**88.68**	92.44
+ HorizonKV	4%	**89.16**	82.50	75.11	73.14	**87.72**	**84.57**	82.05	90.09	82.00	84.00	86.79	**92.77**
+ HeatKV	1%	88.81	82.17	75.55	72.31	**87.84**	**83.98**	82.26	89.83	81.50	84.00	88.05	**93.30**
+ HorizonKV	1%	**88.86**	82.17	**76.86**	**73.55**	86.93	83.79	82.26	**90.11**	**82.00**	84.00	88.05	92.69
Infinity-8B	100%	92.12	88.15	80.79	73.97	89.47	83.79	85.32	93.23	81.00	94.00	89.31	94.74
+ HeatKV	4%	92.32	87.49	**78.60**	**74.79**	89.66	83.59	85.11	93.08	**83.50**	**96.00**	88.05	94.86
+ HorizonKV	4%	**92.37**	**88.15**	78.17	74.38	**89.78**	83.59	**85.74**	**93.38**	80.50	92.00	**88.68**	**94.95**
+ HeatKV	1%	92.07	87.71	**79.91**	72.73	**89.72**	82.81	85.11	93.04	**83.00**	92.00	**88.68**	94.74
+ HorizonKV	1%	**92.42**	**87.93**	79.04	**73.14**	89.66	**83.20**	**86.48**	**93.10**	81.50	**94.00**	86.16	**95.15**

@4% and HorizonKV @4% require 1.876 and 2.147 s/image, respectively. At batch size 16, the corresponding latencies are 1.825 and 1.960 s/image. Relative to HeatKV, the latency ratios are approximately $1.30\times, 1.14\times, 1.07\times$ for batch sizes 1, 4, and 16, respectively. The runtime overhead comes from realizing the selected physical memory states, including `K8V8` state conversion and reconstruction before standard attention computation. Future Demand, Compression Sensitivity, and target-stage allocation are all computed offline, so no online importance estimation or schedule optimization is required. The current implementation prioritizes faithful realization of the physical cache states rather than kernel-level latency optimization. Fused quantization/dequantization, direct low-bit KV attention, and optimized state-transition kernels are complementary system optimizations.

E.2 Runtime Execution

Calibration and schedule optimization are fully offline. Runtime generation only loads the frozen schedule $S = \{S_t\}$ and executes the predetermined state transitions. At target stage t, the runtime reads S_t. For a group assigned to Native, the original historical representation is reused. For a group assigned to `K8V8`, the persistent INT8 payload and FP16 scales are reconstructed when the group is consumed by attention. For a transition `Native` $\rightarrow$ `K8V8`, the `Native` historical state is quantized and replaced by its compressed persistent representation. For a transition to Drop, the group is removed from subsequent historical attention. Runtime therefore does not recollect $A_{g,u}$, recompute $D_{g,u}(q)$, evaluate $C_{g,t}(q)$, or solve the budget-constrained allocation problem. These quantities are used only during offline schedule construction. The runtime path is

$S_t \rightarrow$ load retained history $\rightarrow$ decode `K8V8` states $\rightarrow$ attention $\rightarrow$ apply transitions for S_{t+1}.

HorizonKV is therefore an offline-calibrated, frozen-schedule, runtime-executed KV-cache management framework.