Level of Detail Attention: Allocating Resolution Instead of Sparsity

Anonymous authors
Paper under double-blind review

Abstract

Most sparse attention methods attend to a subset of keys, ignoring the rest and thereby amplifying any mistakes while entirely removing the ignored set from softmax normalization. We introduce Level of Detail (LoD) Attention, a training-free approximation to dense softmax attention that represents less important keys and values at reduced resolution instead of discarding them. For language inference, LoD requires no changes to pretrained model weights and achieves $O(N^{3/2})$ prefill work and $O(N^{1/2})$ work per decode step. Experiments on Qwen3.8 and K2 Horizon show that LoD largely preserves aggregate quality. On Qwen3.8, two-tier LoD accelerates attention computation by over 6x during both prefill and decode in our 128K prompt batch size 8 benchmarks. Its similarity-based grouping scheme also permits the reduction of the LLM key-value cache to a compact INT4 representation, saving VRAM with little additional aggregate quality loss. To show the versatility of our method across modalities we also demonstrate LoD for MiniMax-H3 video generation.

1 Introduction

Transformers attend to one key-value (KV) pair for each context token. This provides reliable recall, but attention work grows with context length. The usual response is sparse attention: estimate which keys or blocks matter and attend only to those. When the estimate is incorrect, important keys get ignored and end up contributing nothing. Even when it is correct, the weight distribution of the remaining keys is distorted because softmax now divides by a smaller denominator. This mismatch can make it difficult to apply attention sparsity at inference time to traditional pretrained model weights.

We propose grouping the key-value entries into a hierarchy of similarity regions, drilling down into important regions, and then attending to a mix of less important region summaries and more important region key-value entries. This way, even an incorrect estimate still provides attendance to important keys, albeit in summary form. Additionally, overall attention mass and the softmax denominator move closer to full attention. We also always attend to a small local region of KVs surrounding the current query.

In LoD, less important similarity regions remain ”closed” with each contributing one KV region summary, while more important regions are ”opened”. In our simplest two-tier form, open regions contribute their entire set of KV pairs. Our recursive three-tier form refines those open regions yet again, opening the most important fixed-capacity page from within. In this form, open pages contribute their entire set of KV pairs, and open regions contribute a summary of their disjoint remainder (with those KV pairs discounted). Figure 1 contrasts pruning with the two-tier rule; the recursive rule is defined in Section 2.

The LoD attention operator is distinct from the user choice of a query-time similarity metric and region constructor. For application to pretrained language models, we form growable regions using online processing similarly to Key-Value Means (KVM) (Goldstein et al., 2026) or Online Vector Quantization (OVQ) (Alonso et al., 2026). In our three-tier mode, we subdivide these regions into pages of up to 16 KV pairs. Our LLM applications do $O(N^{3/2})$ prefill work and $O(N^{1/2})$ work per decode step.

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.

For video, we use a two-tier model with regions as 64-token pages from 4x4x4 spatiotemporal neighborhoods for video and 64-token temporal neighborhoods for audio. The first tier summarizes each of these pages into a mean query summary and mean KV summary, and we choose a subset to open. Our choice of similarity metric is slightly different than for language LoD.

The application to both language and video demonstrates that LoD has uses as a modality-independent attention primitive rather than just a single-purpose cache trick. The two systems share the same softmax normalization rule, but differ in how regions are built and scored. Every key is covered once, either exactly or at reduced resolution through a disjoint aggregate summary. Under the schedules used here, language LoD has $O(N^{3/2})$ total attention work, while hard-capped Video LoD limits optional exact promotions to $O(N^{3/2})$ interactions.

Our video implementation prioritizes practical GPU performance at the evaluated sequence lengths. With fixed 64-token pages, global proxy routing costs $O(N^2/64^2)$, while retaining one summary per closed page costs $O(N^2/64)$. With a hard-cap allocation, optional exact promotions are limited to $cN^{3/2}$ interactions. Consequently, the complete released video operator remains asymptotically quadratic, although its substantially reduced interaction count and GPU-oriented implementation provide practical speedups. A deeper summary and routing hierarchy that groups closed pages could make the video construction subquadratic, but we leave that design to future work.

LoD is a general resolution-allocation principle; subquadratic language attention is one particularly effective scheduling instance, not a required property of every implementation.

Our main contributions are:

- A mathematical definition of level-of-detail attention in which every original KV pair is represented exactly once at an active resolution frontier - the disjoint collection of exact KV entries and closed summaries presented to the final softmax.
- A weight-preserving retrofit for pretrained LLMs, with online semantic regions, region-owned exact pages, and a square-root schedule that gives $O(N^{3/2})$ sequence attention work.
- A recursive three-tier implementation that opens one highest-scoring page per routed region, keeps the disjoint residual summarized, and merges coarse, local, and exact-page branches by log-sum-exp (LSE).
- Region-owned mean-residual INT4 storage that cuts LLM allocated recursive KV-cache storage increasingly with context length and by over 50% by 128K, while recursive routing limits reconstruction costs to the small exact frontier rather than the full history.
- A locality-based video instantiation using the same operator, with spatiotemporal locality, measured denoising speedup, and limited quality comparisons.

2 Level-of-Detail Attention

2.1 KV entries, regions, and pages

A base attention item is one key-value pair (k_i, v_i). We call it a *KV entry*. Let a query q assign attention logit $x_i = s(q, k_i)$, where s includes the model's native scaling and positional transformation. Separate the KV entries into local and remote segments by keeping a block local window around each query. Partition the remote KV entries into disjoint regions $r \in \mathcal{R}$.

For any nonempty set of entries A, define its exact partition mass and value numerator as

$$Z_A(q) = \sum_{i \in A} e^{x_i}, \qquad N_A(q) = \sum_{i \in A} e^{x_i} v_i. \tag{1}$$

Let $n_A = |A|$, $K_A = \sum_{i \in A} k_i$, $V_A = \sum_{i \in A} v_i$, $\bar{k}_A = K_A/n_A$, and $\bar{v}_A = V_A/n_A$. Its zeroth-order coarse approximation is

$$\widetilde{Z}_A(q) = n_A e^{s(q, \bar{k}_A)}, \qquad \widetilde{N}_A(q) = \widetilde{Z}_A(q)\bar{v}_A. \tag{2}$$

Equivalently, the set contributes one summary logit $s(q, \bar{k}_A) + \log n_A$. The $\log n_A$ term is essential: without it, a summary representing 50 KV entries would compete as though it represented one.

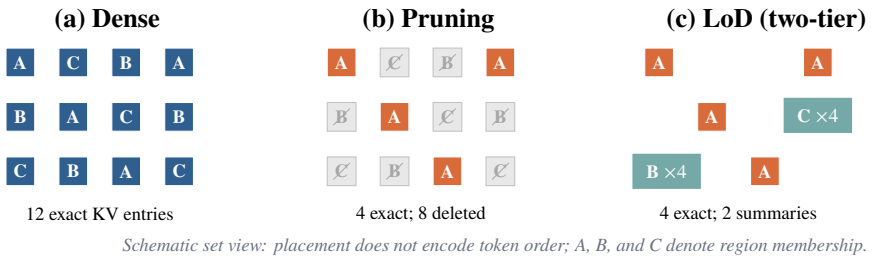

Figure 1: Dense attention reads every KV entry. Pruning deletes entries and renormalizes the survivors. In the two-tier form of LoD, the selected region A is represented by its original KV pairs, while regions B and C each contribute one population-corrected summary. Region members are deliberately shown noncontiguously because LLM regions are semantic rather than positional.

For bounded refinement, each region r may additionally be partitioned into fixed-capacity pages $\mathcal{P}_r$. Pages are disjoint and cover the region. Each page stores its original KV entries plus its own count and key/value sums. In the LLM implementation, a KV entry that ages out of the local segment is first assigned to a semantic region, then given a stable region-local ordinal and packed into that region's pages; entries in a page need not be consecutive in the original sequence, but are added in order.

2.2 Two-tier and recursive three-tier frontiers

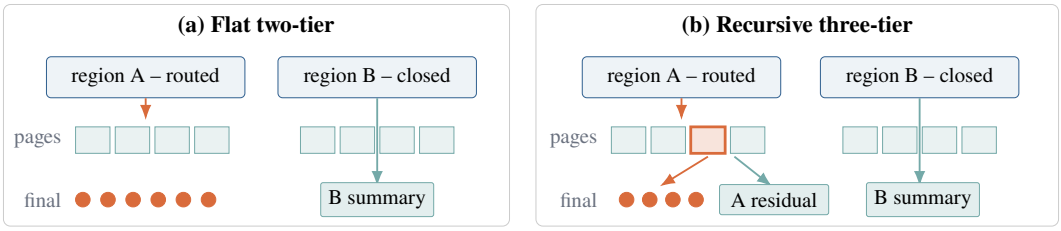

Figure 2: Two-tier and recursive LLM layouts. A *KV entry* is one original key-value pair. Regions own disjoint pages, and pages own disjoint KV entries. In the two-tier form, a routed region is fully exact. In the recursive three-tier form, page summaries are an intermediate routing level: one selected page becomes exact while all other pages of that routed region are represented by one residual summary. The always-exact local field is omitted.

Let $O(q)$ be the open regions selected for refinement, and let $(Z_{\text{loc}}, N_{\text{loc}})$ denote an always-exact local or prefix field. The two-tier operator replaces every open region summary with all of that region's original KV entries:

$$\text{LOD}_{\text{two-tier}}(q) = \frac{N_{\text{loc}} + \sum_{r \notin O(q)} \widetilde{N}_r + \sum_{r \in O(q)} N_r}{Z_{\text{loc}} + \sum_{r \notin O(q)} \widetilde{Z}_r + \sum_{r \in O(q)} Z_r}. \tag{3}$$

A selected region's summary is switched off when its KV entries are switched on, so there is no double counting. Opening every region in this two-tier form recovers dense attention exactly.

Our LLM implementation also provides a recursive three-level variant. After routing to a region r, it scores the summaries of pages inside that region and chooses a single best page

$$p_r^*(q) = \arg\max_{p \in \mathcal{P}_r} \left[s(q, \bar{k}_p) + \log n_p \right]. \tag{4}$$

Only the KV entries in p_r^* receive exact attention. The rest of the routed region remains represented by one disjoint residual summary computed from $K_r - K_{p_r^*}$, $V_r - V_{p_r^*}$, and $n_r - n_{p_r^*}$. Thus

$$\text{LOD}_{\text{three-tier}}(q) = \frac{N_{\text{loc}} + \sum_{r \notin O(q)} \widetilde{N}_r + \sum_{r \in O(q)} \left(N_{p_r^*} + \widetilde{N}_{r \setminus p_r^*} \right)}{Z_{\text{loc}} + \sum_{r \notin O(q)} \widetilde{Z}_r + \sum_{r \in O(q)} \left(Z_{p_r^*} + \widetilde{Z}_{r \setminus p_r^*} \right)}, \tag{5}$$

where an empty residual is omitted. Page summaries are an intermediate routing level; unselected pages are not read exactly and are not individually added to the final softmax. They are represented in combination by the residual summary.

Frontier invariant. Every original KV entry is represented exactly once at the active frontier: directly, or inside exactly one summary whose represented set is disjoint from all exact entries. Dense attention is recovered by refining the complete hierarchy to base KV entries.

2.3 Why high-mass regions should be exact

Write $\bar{x}_r = n_r^{-1} \sum_{i \in r} x_i$. Because $s(q, \cdot)$ is linear in k after the model's native transformations, $s(q, \bar{k}_r) = \bar{x}_r$. The exact regional log-mass decomposes as

$$\log Z_r = \log n_r + \bar{x}_r + J_r, \qquad J_r = \log \mathbb{E}_{i \in r}[e^{x_i}] - \mathbb{E}_{i \in r}[x_i] \geq 0. \tag{6}$$

Thus the summary is a Jensen lower bound on regional mass. It is exact when all regional logits are equal, and becomes conservative as within-region score variance grows. Opening a large estimated-mass region removes an error term that has large leverage on the global denominator. This motivates routing by the coarse log-mass $s(q, \bar{k}_r) + \log n_r$, subject to an exact-work budget. The same Jensen lower bound underlies the centroid-based attention estimation in RetroInfer (Chen et al., 2026b).

Mass alone is not the complete output-error objective. The regional numerator error is

$$N_r - \widetilde{N}_r = (Z_r - \widetilde{Z}_r)\bar{v}_r + \sum_{i \in r} e^{x_i}(v_i - \bar{v}_r). \tag{7}$$

The first term is the missing Jensen mass; the second is a score-value covariance. A region can have modest mass yet contain a decisive value, or have high mass but nearly constant values. We therefore use high estimated mass as a simple, hardware-friendly first criterion, not as a theorem that top mass is always the optimal route objective.

In fact, in Video LoD we use a more complex objective that does take into account value variance as we found that it was helpful there.

2.4 Asymptotic cost

Suppose a length-N history has $M(N)$ semantic regions, an always-exact field of width W, at most K routed regions, pages of capacity P, and a refinement limit B on region population. Two-tier query work is $O((M + W + KB)d)$. Recursive decode additionally scores at most $K\lceil B/P \rceil$ page summaries and reads KP exact entries, giving $O((M + W + K\lceil B/P \rceil + KP)d)$. Prefill fully refines its routed regions but remains bounded by the two-tier expression. We use $M(N) = 16\sqrt{N}$, $B = 1024$, $K = 8$, $P = 16$, and fixed W, giving $O(\sqrt{N})$ work per query and $O(N^{3/2})$ work over causal prefill, including state maintenance. The coarse state is sublinear under the square-root schedule, but this does not imply sublinear memory usage.

3 Semantic LoD for Pretrained Language Models

3.1 Online semantic regions

We replace a pretrained model's global softmax-attention layers without changing or fine-tuning its projections, normalization, positional encoding, output gating, output projection, or any other learned weight. LoD receives post-projection, post-position tensors. The first KV entry is kept separate as a protected sink. Recent KVs remain in an exact local BSWA (Hwang et al., 2024) window. When a chunk ages out, its KV entries update a growable semantic state related to KVM (Goldstein et al., 2026) and OVQ (Alonso et al., 2026): some entries seed new state rows under the current capacity, while the remainder are assigned to existing regions. Each region stores a count and key/value sums.

After region ownership is known, each region's original KV entries are packed into fixed-size pages. Region-local ordinals are stable and preserve chronological order among entries assigned to the same region, but a page is not an arbitrary consecutive block of the physical cache and its members need

not be adjacent in token position. Each region maintains its own page list and every page maintains count and key/value sums, enabling page-level query routing without losing the original KV pairs.

The reference implementation processes prefill in blocks of at most 16,384 query tokens. The first block, and therefore any prompt that fits within one block, is evaluated with exact causal attention. For longer prompts, each subsequent block uses LoD for the earlier remote prefix while retaining the current block and up to 256 immediately preceding KV entries exactly. Thus, a full later block has a maximum exact local field of 16,384 + 256 = 16,640 KV entries; a partial block uses only its actual length plus the available lookback. During decode, the implementation retains up to 512 recent KV entries exactly and incorporates older entries into the semantic state in 256-token updates. The remaining reference settings are eight routed regions, 16-entry pages, and a refinement cap of B = 1024 entries per region (Section 2.4); regions above the cap are never opened and remain count-corrected coarse summaries. Appendix A.5 ablates this cap.

3.2 Architecture-aware state geometry

Naively assigning a KV entry to the state centroid with the largest raw dot product would deliver inconsistent results across model families. Averaging keys changes centroid length through both genuine token scale and directional cancellation. Whether that length is meaningful depends on the model's attention normalization.

For models without explicit key normalization, we do directional assignment to centroids. Writing $u(x) = x/\mathrm{rms}(x)$, the score $u(k)^\top u(c)/d = \cos(k, c)$ is used for assignment and farthest-entry admission; stored key/value sums remain unnormalized. Thus clustering removes token length and centroid-resultant scale without changing final attention logits.

When the attention module explicitly normalizes each key before a fixed learned gain, token-dependent radial scale has already been removed. In that case, the centroid resultant length is a calibrated measure of whether one direction represents the region. We track

$$\rho_r = \frac{\mathrm{rms}(\bar{k}_r)}{n_r^{-1} \sum_{i \in r} \mathrm{rms}(k_i)} \tag{8}$$

and use coherence-aware assignment proportional to $\cos(k, \bar{k}_r)\rho_r$. The resulting policy is layer-local and implicitly specified by the model's attention choices:

1. if the native attention explicitly normalizes K, use coherence-aware assignment;
2. otherwise, use directional assignment.

Routing and state construction are separate. The canonical route logit remains the population-corrected coarse mass in Equation 2. For architectures without explicit QK normalization, raw query magnitude changes the relative strength of $q^\top \bar{k}_r$ and $\log n_r$. We therefore rank by query direction,

$$\mathrm{route}_r(q) = \beta \frac{q^\top \bar{k}_r}{\mathrm{rms}(q)} + \log n_r, \tag{9}$$

where β is the model's native QK scale, while leaving the actual coarse attention logit unchanged. This correction affects which regions are opened, not how closed-region mass is normalized.

3.3 Cache and kernel implementation

The cache owns four objects per replaced layer: the exact local field, coarse state, region ownership metadata, and the archive of original KV entries organized into pages. Our optimized recursive path computes closed-region summaries plus local attention, selects one page within each routed region, adds a disjoint residual summary for the rest of that region, computes exact attention over the selected pages, and merges the branches using their LSEs. This is algebraically equivalent to one softmax over Equation 5.

Page-mean-relative INT4 pages Quantization is applied only after semantic ownership is known. For each key or value vector x_j in a centroid-owned page p, the archive stores a page mean and a signed four-bit residual,

$$\hat{x}_j = \bar{x}_p + s_{p,g}\, Q_4\big((x_{j,g} - \bar{x}_{p,g})/s_{p,g}\big), \tag{10}$$

with four-channel groups and signed codes in $[-7, 7]$. Scales are initialized from the residual range and the release uses an L2 scale refinement. Mean removal narrows the coded range; semantic ownership gives an additional reason for *key* residuals to be small, because the keys were assigned for proximity to the same centroid. We do not assume the same geometric guarantee for values.

Three-tier decode makes this compression cheap to consume. Only one 16-entry page per routed centroid is reconstructed; the rest of history participates through centroid or residual summaries. Exact-page dequantization is therefore $O(KP)$ with $K = 8$ and $P = 16$, rather than $O(N)$. The LoD cache is authoritative: remote semantic leaves replace, rather than shadow, a second chronological full-KV cache. This interaction between semantic ownership, low-bit residuals, and a small exact frontier is a capacity advantage beyond attention arithmetic alone.

3.4 Prefill-decode asymmetry

Prefill deliberately differs across two and three-tier modes. For all modes, up to eight routed centroids containing at most 1,024 leaves are fully refined during prefill, even when the archive is three-tier. Large prefill kernels can process that regular selected set efficiently, while single-page recursion is most valuable for bandwidth-bound autoregressive decode.

We update the region list only once every 16k tokens during prefill, while we update it every 256 tokens during decode. This is a speed optimization that allows us to process prefill tokens efficiently, and allows us to defer most LoD operations until the 16k+ regime where it tends to overcome its extra constant time costs to overtake full attention.

4 LLM Experiments

We evaluate our optimized LLM LoD implementation on Qwen3.8-27B-FP8 (Qwen Team, 2026) and K2-Horizon-32B-FP8 (IFM Team, 2026) on vLLM (Kwon et al., 2023) 0.27.1 using AMD MI325X GPUs. We report ProLong (Gao et al., 2025) prompt loss, LongBench v2 (Bai et al., 2025), RULER (Hsieh et al., 2024), and matched ProLong speed sweeps versus full attention. Appendix A gives the complete per-length results, and additional INT4 memory consumption experiments.

4.1 LLM Prolong perplexity

We evaluate all three LoD modes versus full attention on eight 64k token documents from the Prolong dataset. Perplexity stays within 0.26% of full attention across all models and methods.

Table 1: Perplexity across eight 65,536-token ProLong documents. Parentheses show the relative change from full attention.

Model	Full attention	Two-tier BF16	Three-tier BF16	Three-tier INT4
Qwen3.8-27B	1.557	1.561 (+0.228%)	1.561 (+0.216%)	1.561 (+0.255%)
K2-Horizon-32B	1.642	1.643 (+0.092%)	1.643 (+0.095%)	1.644 (+0.112%)

4.2 LongBench v2 performance

We test Qwen and K2 Horizon on the full 503 example set of LongBench v2 (Bai et al., 2025). We disable model thinking, constrain the answer to A/B/C/D, and cap input at 131,072 tokens by retaining the first and last halves, causing 205 Qwen and 181 K2 prompts to be truncated. We observe comparable performance using LoD versus full attention.

Table 2: LongBench v2

Model	Full attn	Two-tier BF16 ↑	Three-tier BF16 ↑	Three-tier INT4 ↑
Qwen3.8-27B	52.09%	52.09%	53.08%	53.08%
K2-Horizon-32B	41.75%	43.94%	42.35%	41.95%

4.3 RULER PERFORMANCE

We evaluate LoD on RULER (Hsieh et al., 2024) using LM Eval Harness (Gao et al., 2024) at 64k context length in two-tier mode, and find similar aggregate results to full attention: Qwen 94.39 LoD/96.17 full, and K2 Horizon 92.04 LoD/93.06 full. Individual regressions concentrate in NIAH Multikey 2/3 and QA: both models decline on HotpotQA and SQuAD QA, while Qwen alone declines on frequent-words extraction. For full results by category please see Table 3 in Appendix A.

4.4 LLM PREFILL AND DECODE TIMINGS

We evaluate prefill and decode timing using full LoD across all modes at different tensor parallelism and batch sizes. The prefilled source documents are 64k+ token Prolong documents, with multiple appended to reach 128k when necessary. Decode latency is calculated over 1,024 timed post-first-token steps in order to properly reflect amortized state update costs. We perform prefill first and time it separately from the decode phase. To estimate attention-only timing we subtract the time of a run of the full model with attention replaced by a no-op to avoid disturbing cudagraph optimizations. For results, please see Figure 3 below, and Figures 4, 6, and 5 in Appendix A.

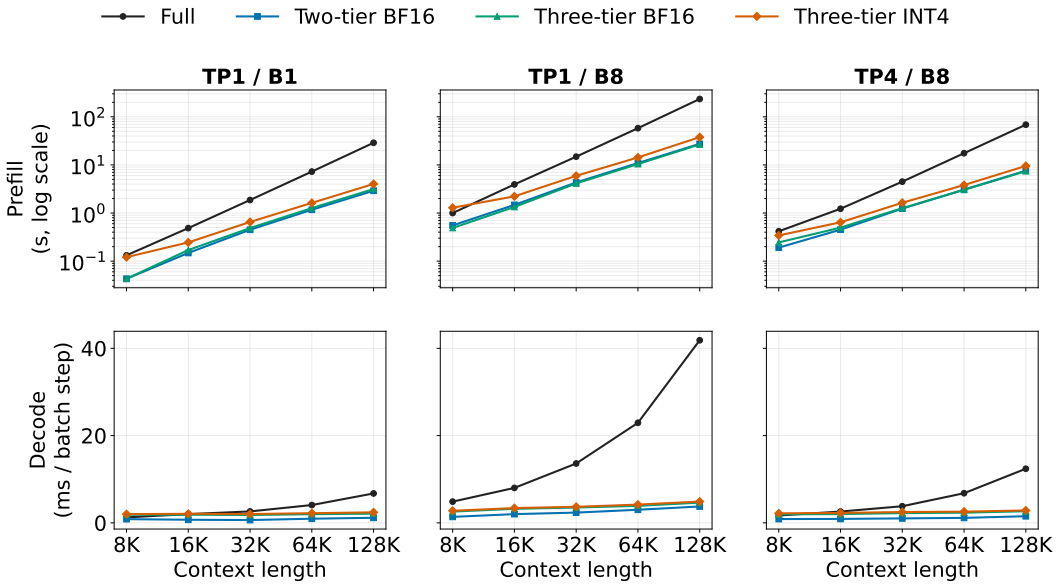

Figure 3: Qwen3.8-27B attention prefill and decode timings

Prefill and decode attention speedups range from 4x to over 13x at 128k token context length on Qwen. Interestingly, the results are more muted for K2 Horizon. We theorize that this is due to differences in head size, GQA ratio, and head count: LoD must do routing and top-8 selection separately for 64 query heads on K2 Horizon versus only 24 for Qwen. With small head-size and more heads, K2's attention is more parallelizable and efficient to compute, so LoD's fixed costs cause speedups to appear in longer context regimes. We may need to optimize the LoD implementation for K2 Horizon in a different manner to unlock further improvement.

We additionally evaluate Qwen with the seven-token `z-lab/Qwen3.8-27B-DFlash2` (Inco AI, 2026) drafter, and recursive three-tier BF16 LoD independently routing every speculative target position. With DFlash2 enabled we find LoD provides similar kinds of speed increases as it did for non-speculatively decoded Qwen. Please see Figure 7 in Appendix A for detailed measurements.

5 Space-Time LoD for Video

The video implementation of LoD leaves the operator the same, but changes the region constructor and scoring formulas. We apply the two-tier method for simplicity. Our application to MiniMax-H3 (Chen et al., 2026a) receives an exact text prefix followed by audio and video tokens. Raster order is permuted into tile-major order so each 64-token page is a compact locality region. Audio pages are 64-tokens long and video pages are $4 \times 4 \times 4$ space-time blocks. Each query page is mapped to one KV page set.

The common component across LLM and video scoring is $q^\top \bar{k} + \log n$. Our video scoring targets approximation error, while LLM LoD primarily targets attention mass. To achieve this, we add explicit query/key cancellation correction, a 1.5 mass exponent, and value variance. See Appendix B for details.

Similarly to the local window used in LLM LoD, we employ a local window akin to Sliding Tile Attention (Zhang et al., 2025b) ("STA-3") for both audio and video. Video queries open a $3 \times 3 \times 3$ neighborhood of video pages, while audio queries open their previous, current, and next audio pages. Temporal cross-modal neighborhoods and the final partial audio page are also mandatory exact routes. We keep full attention between text tokens and audio/video, as a fixed length text description does not scale longer as the resolution or clip length is increased.

For each query page, a fused kernel scores every page summary, and uses a per-head threshold allocator to select approximately the top-k audio and video pages. One online softmax then covers the exact text prefix, exact KV entries of all selected pages, and one summary for every unselected page; a selected page's summary is disabled.

The coefficient c sets an optional remote exact-promotion budget of $cN^{3/2}$ query-key pairs. After fixed text and mandatory local work are accounted for, the allocator assigns each audio/video query an allowance $A = \lfloor cN^{3/2}/N_{\mathrm{AV}} \rfloor$. A hard-cap allocator enforces this optional-promotion budget. For efficiency, we also implement a sampled-threshold allocator with cross-layer budget correction, which targets the same aggregate budget without strict per-layer enforcement. One interaction per closed remote page forms the mandatory coarse floor, and promoting a full 64-token page adds 63 interactions, which is exact for a full page and a conservative charge for a ragged optional page.

On an NVIDIA RTX PRO 6000 Blackwell Server Edition with MiniMax-H3 using FP4 weights but BF16 attention, at fixed 1280×768 resolution, increasing duration from 7.29 to 15.79 seconds raised dense denoising latency from 12.71 to 52.46 s/step (4.13x), while LoD c=16 rose from 4.29 to 10.59 s/step (2.47x). Consequently, LoD's speedup over dense increased from 2.97x to 4.95x. Appendix C reports the timing table. We compare to dense attention and Sol Attention (Li et al., 2026a) using VBench (Huang et al., 2024). We also run a custom frame-aligned comparison against the dense-attention reference, reporting the average cosine similarity between corresponding frames in DINO (Caron et al., 2021) and CLIP (Radford et al., 2021) feature spaces. All tests are run against the same variety of prompts at c=16 and c=64, or equivalent amounts of Sol Attention. We find broadly similar results to Sol Attention in both speed and quality. Please see Appendix C for full Video LoD results.

6 Related Work

Semantic indexing and sparse retrieval. Squeezed Attention clusters fixed-context keys offline and uses centroids to retrieve exact members (Hooper et al., 2025). Saap learns an asymmetric query-side index for semantically clustered keys (Mazaré et al., 2025). Quest, InfLLM, and ClusterKV use query-aware pages, representative memory units, or semantic cache manipulation to retrieve a sparse exact set (Tang et al., 2024; Xiao et al., 2024; Liu et al., 2025). These methods are central comparisons for routing and page layout. Their coarse representations generally act as indexes; unselected regions do not remain as low-resolution operands in the final softmax.

Compressed global context and exact refinement. Native Sparse Attention combines a compressed global branch, selected blocks, and a local window (Yuan et al., 2025). KVM and OVQ build online compressed states but do not preserve a query-recoverable archive of original KV entries as part of the same operator (Goldstein et al., 2026; Alonso et al., 2026). RetroInfer (Chen et al., 2026b)

combines exact attention to a steady set of tokens and retrieved clusters with cluster-size-weighted centroid estimates for unretrieved clusters. Its wave index supports segmented clustering and incremental updates, while its wave buffer manages KV storage and movement across GPU and CPU memory. Double-P similarly combines exact attention to promoted clusters with size-weighted summaries of unpromoted clusters (Ni et al., 2026). LoD shares that exact-or-summary structure. Its differentiating emphasis is an online growable semantic state for post-training conversion, exact local attention, region-owned recursive/quantized pages, architecture-aware state geometry, and a modality-independent frontier definition. We do not claim that mixing exact and approximate regions is unprecedented.

Video and diffusion attention. VSA uses pooled locality tiles to select exact fine attention, but drops unselected tiles (Zhang et al., 2025a). PSA assigns multiple pooled resolutions to query-block pairs and includes population corrections, making it a direct multilevel analogue of LoD (Li et al., 2025); unlike our current two-level video implementation, PSA can also prune its lowest-ranked blocks. PISA replaces keep-or-drop sparsity with exact critical blocks and a Taylor approximation for the remainder (Li et al., 2026b). SVG-EAR uses centroid compensation for skipped blocks and routes exact work by estimated approximation error (Zhou et al., 2026). Sol-Attn (Li et al., 2026a) independently combines exact selected blocks with pooled K/V contributions for unselected blocks in a shared online softmax. Its principal distinctions are activation-dependent density and proxy reuse. These works strongly support the broader exact-or-approximate direction. Our video instance is simpler: one count-corrected mean summary per closed locality page, complete coverage, and a single exact-or-summary frontier shared with the LLM formulation.

Efficient exact attention. FlashAttention reduces IO while preserving dense attention exactly (Dao et al., 2022). LoD is complementary: it changes the represented field before an exact kernel evaluates each branch, then combines branch outputs by exact LSE renormalization.

7 Limitations and Discussion

Systems trade-offs remain. Region ownership, page pools, quantization metadata, routing scratch, and coarse state all consume memory. End-to-end gains depend on context length, attention share, batch shape, hardware, and model-specific optimization. Short video clips and short LLM contexts see little benefit. Kernel portability and interaction with continuous batching require further work.

Video evaluation is limited in scale. We evaluate four prompts, one seed, and two attention budgets; these results demonstrate cross-modal transfer and practical performance but are not population-level estimates of quality parity. We treat video as evidence that the operator transfers, not as evidence that quality is maintained at a specific level of retained attention.

8 Conclusion

Level-of-Detail Attention replaces the binary choice of key value sparsity with a choice of region resolution instead. Count-corrected summaries retain one exact or approximate contribution for every key in the shared normalization. The two-tier form substitutes all original KV pairs for a selected region summary; the recursive three-tier form instead opens the top-1 page and keeps the disjoint residual coarse. This offers a smooth choice, as refining the complete hierarchy exactly recovers dense attention. Online semantic regions make this a weight-preserving retrofit for long-context LLMs with subquadratic attention work under a square-root schedule. They also make low-bit entry storage natural: entries are compressed as residuals around coherent page means and reconstructed only when their page is opened, yielding over 50% allocated-cache savings in the current 128K sets of experiments without a full-cache decompression pass. Space-time pages show that the operator does not depend on semantic clustering. Our Qwen3.8 and K2 Horizon results show that the systems advantage grows with context, and can result in large attention-subsystem speedups and smaller but significant end-to-end gains even versus a highly optimized vLLM implementation. By allowing attention to allocate resolution instead of ignoring unselected context, we can achieve subquadratic LLM attention scaling and deep KV compression, even without training.

9 AI Usage Disclosure

In this work, we used generative AI tools to formulate mathematical claims, propose and refine hypotheses, design and provide feedback on research methodology or experiments, implement methods, and map datasets.

We also used them to generate figures, suggest experimental parameters, create and edit software code and artifacts, create draft sections, search for information, and for general discussion across a broad swath of topics.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Nick Alonso, Tomas Figliolia, and Beren Millidge. Online vector quantized attention, 2026. URL https://arxiv.org/abs/2602.03922.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, October 2021.

Danze Chen, Zeqing Wang, Ziyue Lin, Xingyi Yang, and Yeying Jin. H3-world: Turning language understanding into world control, 2026a. URL https://arxiv.org/abs/2609.01560.

Yaoqi Chen, Jinkai Zhang, Baotong Lu, Qianxi Zhang, Chengruidong Zhang, Jing Liu, Jingjia Luo, Di Liu, Huiqiang Jiang, Qi Chen, Bailu Ding, Xiao Yan, Jiawei Jiang, Chen Chen, Mingxing Zhang, Cheng Li, Yuqing Yang, Fan Yang, and Mao Yang. Retroinfer: A vector storage engine for scalable long-context llm inference. *Proceedings of the VLDB Endowment*, 19 (5):1016–1031, January 2026b. ISSN 2150-8097. doi: 10.14778/3796195.3796212. URL http://dx.doi.org/10.14778/3796195.3796212.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7376–7399, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Daniel Goldstein, Navneel Singhal, and Eugene Cheah. Key-value means: Transformers with expandable block-recurrent compressed memory, 2026. URL https://arxiv.org/abs/2605.09877.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length LLM inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32631–32652, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1568. URL https://aclanthology.org/2025.acl-long.1568/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. TransformerFAM: Feedback attention is working memory, 2024. URL https://arxiv.org/abs/2404.09173.

IFM Team. Introducing k2 horizon: Frontier performance, radically open, 2026. URL https://ifm.ai/blog/k2/.

Inco AI. DFlash 2: Keep Drafting Parallel, August 2026. URL https://inco.ai/blog/dflash2/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, and Song Han. Sol-attn: Accelerating video generation inference via on-the-fly attention sparsification, 2026a. URL https://arxiv.org/abs/2607.24027.

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers, 2026b. URL https://arxiv.org/abs/2602.01077.

Xiaolong Li, Youping Gu, Xi Lin, Weijie Wang, and Bohan Zhuang. Psa: Pyramid sparse attention for efficient video understanding and generation, 2025. URL https://arxiv.org/abs/2512.04025.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. In *Proceedings of the 62nd Annual ACM/IEEE Design Automation Conference*, DAC '25. IEEE Press, 2025. ISBN 9798331503048. doi: 10.1109/DAC63849.2025.11132479. URL https://doi.org/10.1109/DAC63849.2025.11132479.

Pierre-Emmanuel Mazaré, Gergely Szilvasy, Maria Lomeli, Francisco Massa, Naila Murray, Hervé Jégou, and Matthijs Douze. Inference-time sparse attention with asymmetric indexing, 2025. URL https://arxiv.org/abs/2502.08246.

Wentao Ni, Kangqi Zhang, Zhongming Yu, Oren Nelson, Mingu Lee, Hong Cai, Fatih Porikli, Jongryool Kim, Zhijian Liu, and Jishen Zhao. Double-p: Hierarchical top-p sparse attention for long-context llms, 2026. URL https://arxiv.org/abs/2602.05191.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=KzACYw0MTV.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119638–119661. Curran Associates, Inc., 2024. doi: 10.52202/079017-3801. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 152509–152534. Curran Associates, Inc., 2025a. doi: 10.52202/085713-5099. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=U74MOXPEJd.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, Haocheng Xi, Jintao Zhang, Huanzhi Mao, Joseph E. Gonzalez, Kurt Keutzer, Ion Stoica, and Alvin Cheung. Svg-ear: Parameter-free linear compensation for sparse video generation via error-aware routing, 2026. URL https://arxiv.org/abs/2603.08982. Accepted to ECCV 2026.

A Full LLM LoD Results

A.1 RULER

We run the full RULER benchmark using LM Eval Harness with chat templates on. The results are in Table 3 below.

Table 3: RULER

Model	Task	Full Attention ↑	Two-tier LoD ↑
Qwen3.8-27B-FP8			
	NIAH single 1	100.00	100.00
	NIAH single 2	100.00	99.60
	NIAH single 3	100.00	99.80
	NIAH multikey 1	100.00	99.00
	NIAH multikey 2	100.00	96.20
	NIAH multikey 3	100.00	92.60
	NIAH multiquery	100.00	99.90
	NIAH multivalue	100.00	99.15
	Common-words extraction	99.98	99.90
	Frequent-words extraction	99.40	98.87
	HotpotQA	72.00	67.20
	SQuAD QA	78.85	74.88
	Variable tracking	100.00	99.96
	Mean over all 13 tasks	96.17	94.39
K2-Horizon-32B-FP8			
	NIAH single 1	100.00	100.00
	NIAH single 2	100.00	100.00
	NIAH single 3	100.00	100.00
	NIAH multikey 1	100.00	99.60
	NIAH multikey 2	99.00	93.00
	NIAH multikey 3	100.00	98.20
	NIAH multiquery	100.00	100.00
	NIAH multivalue	97.70	95.95
	Common-words extraction	92.10	94.90
	Frequent-words extraction	85.33	85.40
	HotpotQA	60.40	57.40
	SQuAD QA	75.30	72.08
	Variable tracking	100.00	100.00
	Mean over all 13 tasks	93.06	92.04

A.2 LLM Prefill and Decode Timings

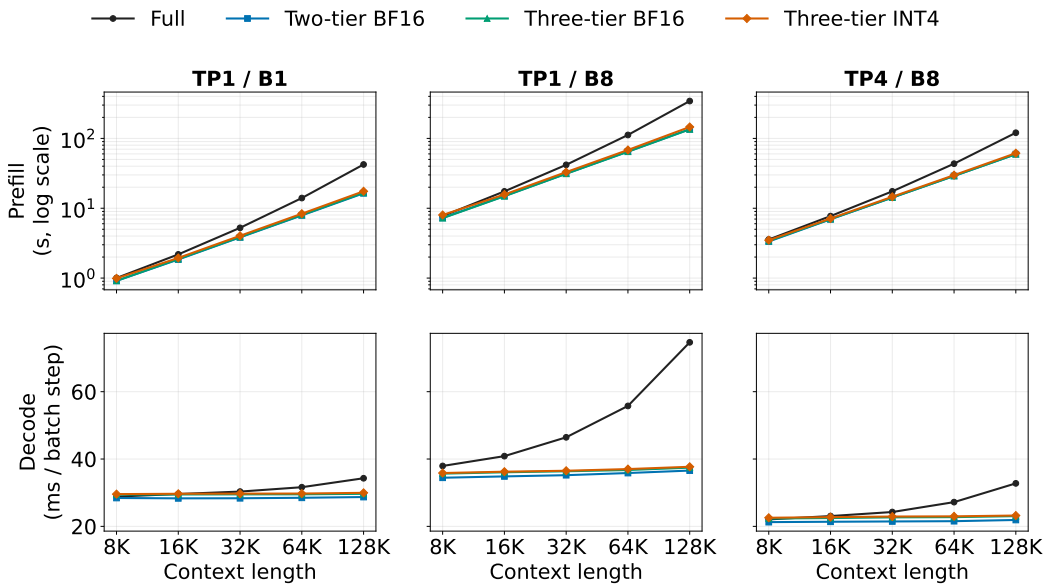

Figure 4: Qwen3.8-27B end-to-end prefill and decode timings

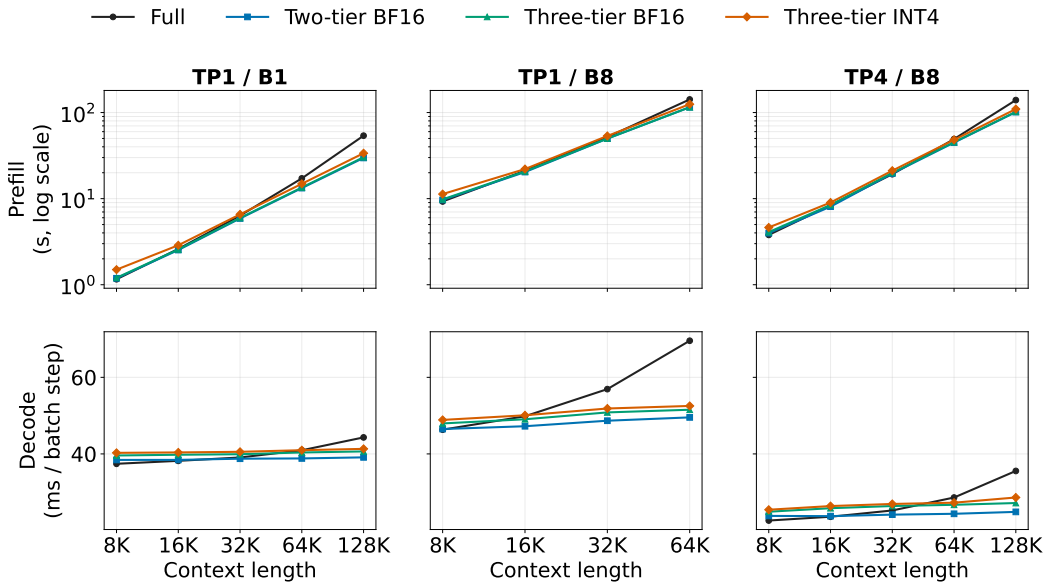

Figure 5: K2-Horizon-32B-FP8 end-to-end prefill and decode timings

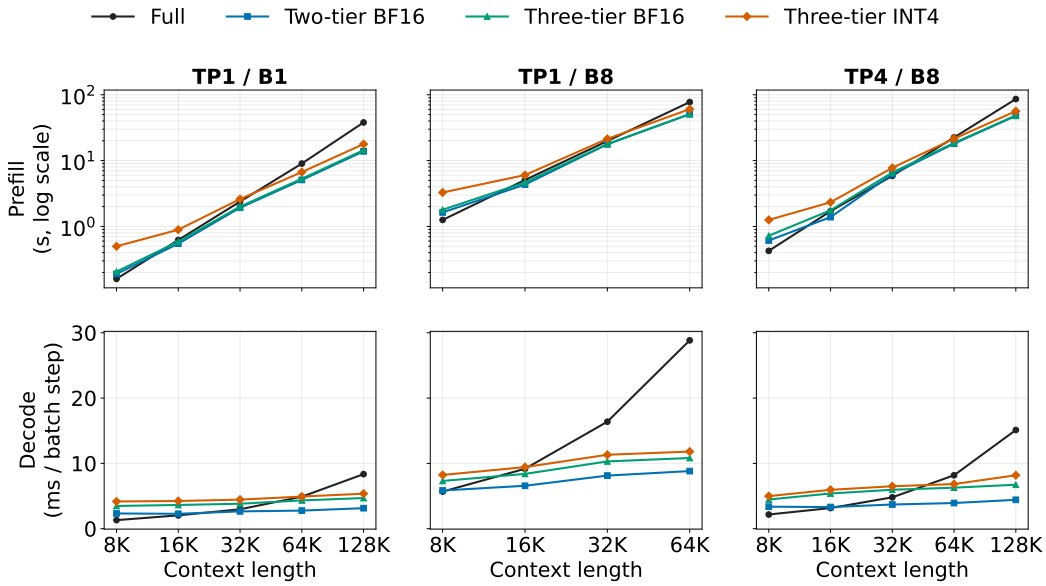

Figure 6: K2-Horizon-32B-FP8 attention prefill and decode timings

A.3 Qwen3.8-27B: recursive LoD with DFlash2

End-to-end decode depends on DFlash2 acceptance, so we additionally provide complete target-verifier cycle timing as well as the average number of tokens emitted per speculative verification cycle.

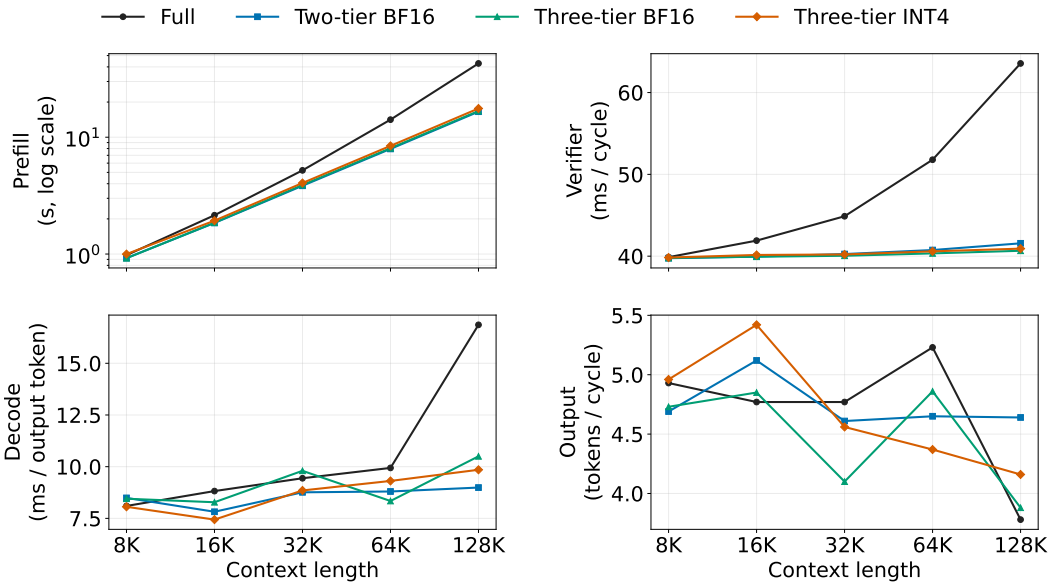

Figure 7: Qwen3.8-27B-DFlash2 end-to-end prefill, decode, speculative cycle wall time, and pooled mean output timings. Measured using Tensor Parallelism 1, Batch 1.

A.4 INT4 Memory Usage

We evaluate cache memory usage for both Qwen and K2 in three-tier INT4 mode versus full attention. Results are shown in Table 4. With no overhead the theoretical maximum savings is 75%, and future

work may be able to approach that by optimizing storage methods for the LoD regions and other overhead.

Table 4: Three-tier INT4 cache memory usage vs. full attention. Persistent attention-cache capacity allocation for one request, with configured capacity equal to the listed context. LoD includes summary, scale, and index storage; both LoD and Full exclude weights, workspaces, and unchanged recurrent state. BF16 payload is calculated from geometry; LoD allocation is measured.

Model	Context	Full BF16 K/V	Three-tier INT4	Reduction ↑
Qwen3.8-27B				
	8K	0.500 GiB	0.430 GiB	13.9%
	16K	1.000 GiB	0.685 GiB	31.5%
	32K	2.000 GiB	1.103 GiB	44.9%
	64K	4.000 GiB	1.925 GiB	51.9%
	128K	8.000 GiB	3.413 GiB	57.3%
K2-Horizon-32B				
	8K	2.000 GiB	2.207 GiB	-10.3%
	16K	4.000 GiB	3.504 GiB	12.4%
	32K	8.000 GiB	5.460 GiB	31.7%
	64K	16.000 GiB	9.145 GiB	42.8%
	128K	32.000 GiB	15.680 GiB	51.0%

At 128K, K2 Horizon's 15.463 GiB of unique three-tier INT4 tensor payload (15.680 GiB allocated) breaks down as shown in Table 5.

Table 5: K2 Horizon 128K three-tier INT4 cache memory usage breakdown

Component	GiB	Share
Packed 4-bit leaf K/V	8.016	51.8%
Quantized page summaries	1.721	11.1%
Page-summary scales	0.860	5.6%
Leaf scales	0.860	5.6%
Centroid sum K/V	1.438	9.3%
Attention-ready centroid/local arena	1.625	10.5%
Inline page directory	0.180	1.2%
Page indices	0.430	2.8%
Overflow hash	0.250	1.6%
Other persistent tensors	0.083	0.5%

A.5 Compact Cap Ablation

Table 6 compares capped and uncapped two-tier LoD with all other settings fixed.

Table 6: Compact cap two-tier ablation study

Metric	Model	Unbounded	Cap 1024	Change
Prolong				
	Qwen3.8-27B-FP8	1.560810	1.560940	+0.0084%
	K2-Horizon-32B-FP8	1.643203	1.643435	+0.0141%
Longbench v2				
	Qwen3.8-27B-FP8	53.88%	52.09%	-1.79 pp
	K2-Horizon-32B-FP8	41.55%	43.94%	+2.39 pp

B Video LoD Details

B.1 Scoring during Routing

Video LoD constructs regions geometrically, but uses query-key similarity to decide which key pages should be represented exactly. For each attention head, let g denote a query block containing $n_g \le 64$ valid query vectors, and let p denote a key page containing $n_p \le 64$ valid key-value vectors. Padding is excluded from both counts. Define the block and page means as

$$\bar{q}_g = \frac{1}{n_g}\sum_{i\in g} q_i, \qquad \bar{k}_p = \frac{1}{n_p}\sum_{j\in p} k_j, \qquad \bar{v}_p = \frac{1}{n_p}\sum_{j\in p} v_j. \tag{11}$$

Here $q_i, k_j, v_j \in \mathbb{R}^{128}$ are the transformed vectors supplied to attention. We suppress the batch and attention-head indices; all quantities below are computed independently for each batch item and head. The mean query $\bar{q}_g$ is used only as a shared routing proxy for the query block. Exact attention still evaluates the individual query vectors.

Averaging vectors can shorten their norms when differently directed components cancel. To distinguish directional cancellation from genuinely small constituent vectors, we define the norm-restoration factors

$$r_q = \frac{n_g^{-1}\sum_{i\in g}\|q_i\|_2}{\max\left(\|\bar{q}_g\|_2, \epsilon_{\text{norm}}\right)}, \qquad r_k = \frac{\max\left(n_p^{-1}\sum_{j\in p}\|k_j\|_2, \epsilon_{\text{norm}}\right)}{\max\left(\|\bar{k}_p\|_2, \epsilon_{\text{norm}}\right)}, \tag{12}$$

where $\epsilon_{\text{norm}} = 10^{-12}$. Except when the denominator is clamped, $r_q\bar{q}_g$ and $r_k\bar{k}_p$ retain the directions of the mean vectors while restoring their norms to the average norms of their constituent vectors.

We define the cancellation-corrected coarse log-mass proxy

$$M(g,p) = \frac{(r_q\bar{q}_g)^\top(r_k\bar{k}_p)}{\sqrt{128}} + \log(n_p r_k). \tag{13}$$

The first term is the corrected similarity between the query-block and key-page representatives. The population term $\log n_p$ accounts for the number of keys represented by the page, while the additional $\log r_k$ incorporates the key cancellation correction into its effective coarse mass. This quantity is a heuristic proxy for the aggregate block-to-page attention mass; it is not the exact log-sum-exp of all constituent query-key logits. $\log r_k$ is also included in the final closed-page bias; r_q and multiplication of the routing dot product by r_k are routing-only.

Attention mass alone does not determine the error introduced by replacing a page with its mean value. We therefore measure the page's value dispersion using the trace of its population covariance:

$$\operatorname{Var}(V_p) = \frac{1}{n_p}\sum_{j\in p}\left\|v_j - \bar{v}_p\right\|_2^2 = \sum_{d=1}^{128}\left[\frac{1}{n_p}\sum_{j\in p}\left(v_{j,d} - \bar{v}_{p,d}\right)^2\right]. \tag{14}$$

Thus $\operatorname{Var}(V_p)$ is the sum of the channelwise population variances, not an unbiased sample variance. It is computed separately for each attention head. A page whose values are nearly constant can be represented accurately by its mean, whereas a page with heterogeneous values has greater potential to benefit from exact refinement.

The final routing priority is

$$S_{\text{video}}(g,p) = 1.5\, M(g,p) + \log\left(\max\left(\operatorname{Var}(V_p), \epsilon_{\text{var}}\right)\right), \tag{15}$$

where $\epsilon_{\text{var}} = 1.17549435 \times 10^{-38}$, the smallest positive normal FP32 value. Equivalently, up to a monotonic logarithm, pages are ranked by

$$\widehat{\text{mass}}(g,p)^{1.5}\max\left(\operatorname{Var}(V_p), \epsilon_{\text{var}}\right), \qquad \widehat{\text{mass}}(g,p) = e^{M(g,p)}. \tag{16}$$

This score is a hardware-friendly heuristic for the benefit of replacing a coarse page summary with its exact keys and values; it is not claimed to be the complete query-conditioned output error.

The implementation stores the query and key means in BF16 and accumulates the norm corrections and value variance in FP32. It evaluates the equivalent score using base-two logarithms and clamps small denominators and variances as shown above. Changing the logarithm base does not alter the page ordering when all terms are converted consistently.

B.2 Quality coefficient

We do not specify the k used for our desired top-k directly. We derive it from the attention-pair budget set by quality coefficient c.

For each 64-query audio/video block:

1. We first install mandatory exact pages. This includes video STA3, audio STA3, all temporally aligned cross-modal pages, and the final partial audio page. These do not consume the routed promotion budget.

2. We next represent every remaining remote page coarsely. Each closed 64-key page initially costs one query–centroid interaction.

3. If a query block has R_g remote pages, its coarse floor costs R_g entries. Promoting one page from its centroid to all 64 keys adds exactly 63 entries. Therefore,

$$k = \left\lfloor \frac{c\sqrt{N_{\mathrm{AV}}}}{63} \right\rfloor$$

where N_{AV} is the number of audio and video tokens.

4. We now have k that we can use with a hard-cap allocator by sorting the pages by their scores and promoting the top-k. For efficiency, we also implement a sampled-threshold allocator with cross-layer budget correction: it estimates a score cutoff from 4,096 deterministic optional query-page samples and uses cross-layer feedback to match the aggregate promotion target, which is $\sum_g \min(k, R_g)$,. Consequently, the number of promoted pages may vary between query blocks. Any pages whose corrected mass-plus-value-variance score lies above the threshold are promoted to full detail. We commonly experienced around a 5% slowdown in denoising speed when using the hard-cap allocator.

C Video LoD Results

We use the NVFP4 quantized weights for MiniMax-H3 and compare LoD outputs with Sol and dense attention at the same settings. We use BF16/CuTe SM120 NVLabs Sana sol-engine branch commit 8e0db4fa562d727ea28b8d63015c196db7d97cae. All prompts are sampled at seed 12345, 1280×768 resolution, 175 frames unless otherwise specified, 50 diffusion steps except Boat, which is seed 12345, 1024x768 resolution, 311 frames, and 20 diffusion steps. We set Sol's τ variable to ensure approximately the same amount of attention is used across LoD and Sol models, based on the LoD c setting. We set Sol to use BF16 QK, PV, and coarse attention with FP32 accumulation. We use no FBC, dense-start steps, or previous-step statistics. We set token_aug=0, diagonal thresholding. Full text attention and sparse audio. The Starship, Dancer, and Mechanics prompts come from Nvidia's Sol Engine for MiniMax-H3 website `https://nvlabs.github.io/Sana/Sol-Engine/H3/`

Prompts:

Starship

Cinematic push-in on a starship bridge: a captain in a high-collared navy tunic stands silhouetted at the observation window as a dreadnought armada charges its hyperdrives. A blinding flash, the bridge shudders, and the fleet is gone.

Dancer

A flamenco dancer spins fast on a wooden stage under a single warm spotlight, her red skirt flaring wide, heels striking the boards in sharp rapid bursts, a guitar driving underneath and hands clapping on the offbeat, camera circling with her.

Mechanics

Two mechanics in a garage at night lean over an open engine bay. The older one wipes his hands and says “It’s not the pump.” The younger one answers “Then what is it?” A radio murmurs in the corner, a wrench clatters onto the concrete, fluorescent tubes buzz overhead.

Boat

integrated_multimodal_description: [Shot 1] A single continuous live-action shot of a small red wooden sailboat floating on a calm pond in a sunlit garden. The camera is stationary at water level, looking across the pond. The boat has a cream triangular sail and a dark wooden mast. It slowly drifts from left to right throughout the shot. Small concentric ripples spread from its hull. Reflections of green willow leaves shimmer gently on the water. The red paint, wood grain, fabric sail and ripples remain sharply visible. Soft natural afternoon sunlight, realistic colors and motion, medium close-up. No people, lettering, cuts or camera movement.
overall_soundscape: Quiet water gently lapping against the small wooden hull, a soft breeze rustling distant willow leaves, and a few faint bird chirps. No speech.
non_diegetic_music: None.

Table 7: Video LoD VBench

Prompt	Attn	c equivalent	Subject consistency	Background consistency	AMT motion smoothness	Aesthetic quality	Imaging quality
Starship	Dense	∞	0.893197	0.924270	0.991669	0.654067	56.7572
	LoD	16	0.839799	0.927810	**0.987995**	**0.618808**	**60.4497**
	Sol	16	**0.879203**	**0.951966**	0.987180	0.534656	53.6701
	LoD	64	**0.855551**	0.916341	**0.991836**	**0.634905**	**57.3854**
	Sol	64	0.853396	**0.943052**	0.990233	0.607397	54.4782
Mechanics	Dense	∞	0.990558	0.977106	0.996943	0.682821	77.6078
	LoD	16	0.987879	0.983610	**0.995666**	0.667480	65.5043
	Sol	16	**0.988655**	**0.984275**	0.995326	**0.682173**	**65.9876**
	LoD	64	0.989721	0.974463	0.996933	**0.717753**	74.1687
	Sol	64	**0.992579**	**0.984999**	**0.997010**	0.675868	**75.3590**
Dancer	Dense	∞	0.835701	0.923646	0.990344	0.614466	60.3605
	LoD	16	**0.849628**	**0.951011**	**0.978198**	**0.605372**	57.9578
	Sol	16	0.830238	0.931357	0.972222	0.602629	**60.0304**
	LoD	64	**0.839046**	**0.935635**	0.979990	**0.586249**	50.7615
	Sol	64	0.818861	0.934604	**0.981581**	0.573794	**62.3608**
Boat	Dense	∞	0.885720	0.930339	0.994445	0.656681	58.6513
	LoD	16	0.893688	**0.945617**	0.992406	**0.691831**	64.6896
	Sol	16	**0.921653**	0.926657	**0.992423**	0.670192	**66.4908**
	LoD	64	**0.879390**	0.924774	0.992015	0.645682	66.4601
	Sol	64	0.829222	**0.931478**	**0.993625**	**0.647736**	**69.2625**



Figure 8: Qualitative comparison on the Starship prompt. Columns show aligned frames at 10%, 50%, and 90% of the video; rows compare dense BF16 attention, Video LoD at coefficient $c = 16$, and pair-matched BF16 Sol attention. Video LoD evaluates 12.2987% of dense pair-equivalent attention, compared with 12.2948% for Sol.



Figure 9: Qualitative comparison on the motion-intensive Dancer prompt. Columns show aligned frames at 10%, 50%, and 90% of the video; rows compare dense BF16 attention, Video LoD at coefficient $c = 16$, and pair-matched BF16 Sol attention. Video LoD evaluates 12.2640% of dense pair-equivalent attention, compared with 12.2586% for Sol.

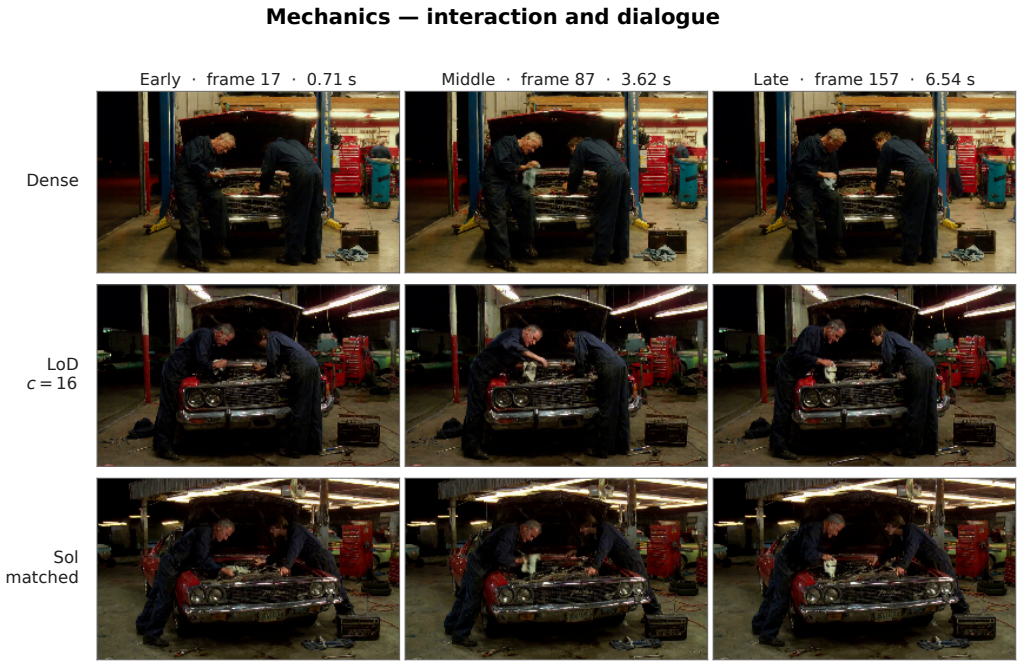

Figure 10: Qualitative comparison on the Mechanics prompt. Columns show aligned frames at 10%, 50%, and 90% of the video; rows compare dense BF16 attention, Video LoD at coefficient $c = 16$, and pair-matched BF16 Sol attention. Video LoD evaluates 12.2952% of dense pair-equivalent attention, compared with 12.3111% for Sol.

Table 8: Video LoD custom frame-aligned similarity on DINO and CLIP

Prompt	Attn	c equivalent	DINO mean	DINO worst 10%	CLIP mean	CLIP worst 10%
Starship	LoD	16	0.620906	0.461145	0.850009	0.799479
	Sol	16	**0.683981**	**0.592460**	**0.866216**	**0.817414**
Starship	LoD	64	0.682222	0.613587	**0.903632**	**0.869562**
	Sol	64	**0.714520**	**0.659436**	0.903249	0.841328
Mechanics	LoD	16	**0.780892**	**0.761560**	**0.857145**	**0.827693**
	Sol	16	0.668671	0.655182	0.792735	0.757200
Mechanics	LoD	64	0.914644	0.906506	**0.943961**	**0.912126**
	Sol	64	**0.920361**	**0.910375**	0.917834	0.876747
Dancer	LoD	16	0.637327	**0.521716**	**0.870477**	**0.813411**
	Sol	16	**0.651783**	0.513389	0.860865	0.783104
Dancer	LoD	64	0.633619	0.528246	**0.858110**	**0.791336**
	Sol	64	**0.688680**	**0.590635**	0.852634	0.782058
Boat	LoD	16	**0.819582**	**0.765188**	**0.926967**	**0.869047**
	Sol	16	0.788946	0.708722	0.907683	0.810188
Boat	LoD	64	0.847989	0.815507	0.906562	0.789498
	Sol	64	0.**873320**	**0.835806**	**0.915267**	**0.831082**

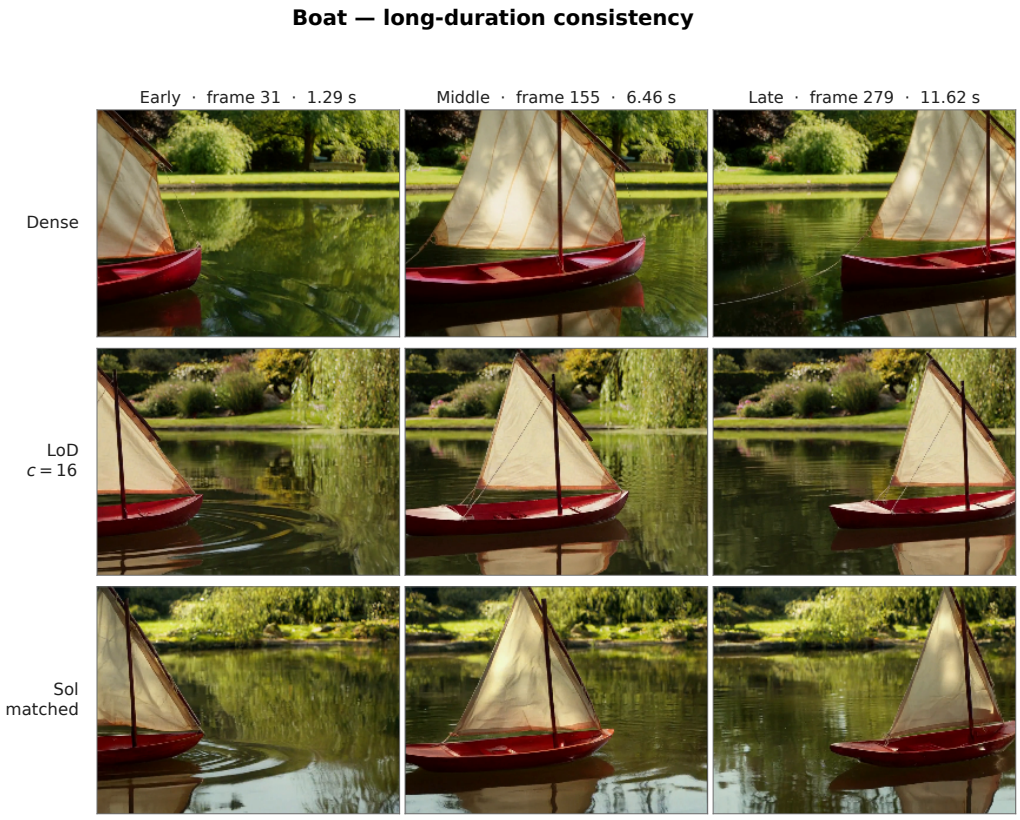

Figure 11: Qualitative comparison on the longer Boat prompt. Columns show aligned frames at 10%, 50%, and 90% of the video; rows compare dense BF16 attention, Video LoD at coefficient $c = 16$, and pair-matched BF16 Sol attention. Video LoD evaluates 10.4369% of dense pair-equivalent attention, compared with 10.4095% for Sol.

Table 9: Video LoD Speed

Prompt	Attn	c equivalent	pair-equivalent attention	Seconds/step	Speedup vs dense
Starship	Dense	∞	100.0000%	12.709	1.00x
	LoD	16	12.2987%	4.285	2.97x
	Sol	16	12.2948%	4.269	2.98x
	LoD	64	33.5783%	6.710 s	1.89×
	Sol	64	33.5865%	6.530 s	1.95×
Mechanics	Dense	∞	100.000%	12.728	1.00×
	LoD	16	12.295%	4.286	2.97×
	Sol	16	12.311%	4.224	3.01×
	LoD	64	33.576%	6.718	1.89×
	Sol	64	33.585%	6.465	1.97×
Dancer	Dense	∞	100.000%	12.691	1.00×
	LoD	16	12.264%	4.274	2.97×
	Sol	16	12.259%	4.235	3.00×
	LoD	64	33.552%	6.708	1.89×
	Sol	64	33.542%	6.488	1.96×
Boat	Dense	∞	100.0000%	24.0030	1.00x
	LoD	16	10.4369%	6.5070	3.69x
	Sol	16	10.4095%	6.3397	3.79x
	LoD	64	28.2729%	10.6336	2.26x
	Sol	64	28.2578%	10.1470	2.37x

Table 10: Video LoD Starship Timing Sweep across clip lengths

Frames	Attn	c equivalent	pair-equivalent attention	Seconds/step	Speedup vs dense
175	Dense	∞	100%	12.709	1x
175	LoD	16	12.2987%	4.285	2.97x
175	Sol	16	12.2948%	4.269	2.98x
243	Dense	∞	100%	22.952	1x
243	LoD	16	10.3447%	6.234	3.68x
243	Sol	16	10.3359%	6.240	3.68x
379	Dense	∞	100%	52.458	1x
379	LoD	16	8.1756%	10.587	4.95x
379	Sol	16	8.1750%	10.708	4.90x