

Forget Without Compromise: Nexus Sampling for Streaming KV-Cache Eviction Under Fixed Budgets

Duc Duong^{1*} Hoang Anh Duy Le^{2,4*} Jianwen Xie³
Anshumali Shrivastava² Zhaozhuo Xu⁴
¹Grinnell College ²Rice University ³Lambda ⁴Workato

Abstract

Long-context and agentic LLM workloads push the KV cache past any fixed memory budget, forcing the inference stack to permanently *evict* tokens at every step of a continuous-inference stream. Existing methods all share the same template: a per-step direct-attention score followed by *deterministic top-K* selection, which converts a single below-cutoff step into an irreversible verdict and permanently erases any subtly important token that direct attention cannot single out from noise. To address this challenge, we propose **Nexus Sampling**, a training-free eviction method that pairs *Nexus scoring*, an iterative walk over direct attention that surfaces bridge tokens, with *weighted reservoir sampling*, which retains tokens with inclusion probability in place of deterministic top- K . Theoretically, we show that Nexus Sampling dominates deterministic top- K in long-run survival of subtly important tokens. Empirically, at 80% KV cache eviction, Nexus Sampling matches dense attention within ~ 1 point on LongBench while outperforming top- K baselines on retrieval-heavy tasks, with up to $10\times$ smaller per-sequence cache memory. Our code is available at <https://github.com/duongduc388222/NexusSampling>.

1 Introduction

GPU memory imposes a hard ceiling on what an LLM inference stack can hold during long-context generation, yet the Key-Value (KV) cache grows linearly with context length and quickly exceeds this budget (Pope et al., 2023; Dao et al., 2022). The gap widens further in long-lived, agent-style deployments such as multi-turn assistants, persistent reasoning loops (OpenAI, 2024), and repository-scale coding agents (Anthropic, 2025; Jimenez et al., 2024), where the effective context

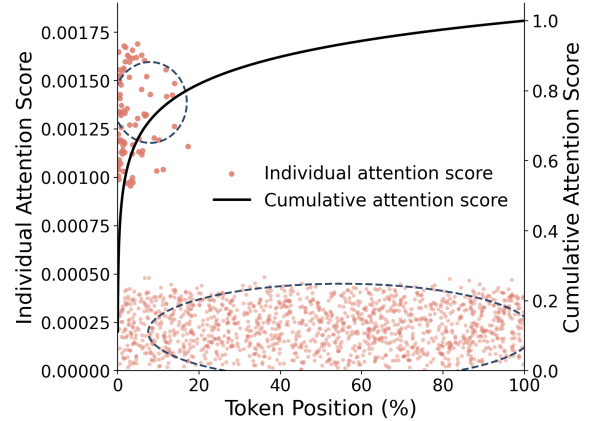


Figure 1: Attention scores are heavy-tailed: $\sim 80\%$ of cumulative mass concentrates on the first $\sim 15\%$ of token positions (black curve), with a small head of high-score tokens (top-left cluster) and a long, noisy tail of marginal-score tokens (bottom band) where deterministic top- K eviction is forced to choose between near-identical scores.

grows toward an unbounded stream. Sparse attention methods (Tang et al., 2024; Le et al., 2026a; Yuan et al., 2026) address this efficiency problem partially by *reading* a still-full cache more efficiently. But once the cache no longer fits in memory, reading it efficiently is not enough: the inference stack must permanently *evict* tokens. A growing body of work (Zhang et al., 2023; Xiao et al., 2024; Cai et al., 2025; Feng et al., 2025; Li et al., 2024; Ghadia et al., 2025) performs this *KV cache eviction*, retaining the tokens that score highest under the current (or recent) queries’ attention and dropping the rest. Existing methods all share the same per-step design as sparse attention: at every eviction step, they retain the K highest-scoring tokens by *deterministic top-K* selection. We argue, however, that **eviction is fundamentally different from sparse attention**: while sparse attention performs a one-time read of a still-full cache, eviction is a *streaming, fixed-budget* problem in which tokens arrive continuously, evictions are irreversible, and the policy must repeatedly decide what to re-

*Equal contributions. Contact: el72@rice.edu and zhaozhuo.xu@workato.com

tain against future queries it has not yet seen. Under this view, per-step deterministic top- K is a suboptimal primitive: it treats each step in isolation and inherently confuses transient marginality with permanent unimportance, so a token whose score is only intermittently above the cutoff is dropped on the first marginal step, even though its long-run average importance across steps may be high.

Deterministic top- K misses tokens whose importance fluctuates. Per-step importance scores in KV caches are noisy and time-varying. Attention in modern LLMs is heavy-tailed (Zhang et al., 2023; Le et al., 2026a): a small head of tokens holds most of the attention mass, while the rest of the cache sits at near-uniform low scores where the relative ranking between tokens is dominated by noise (Figure 1).

A token important to query t may look marginal at $t+1$ and important again at $t+10$. Top- K , however, treats each of these per-step scores as a final verdict: a token that lands below the cutoff at any single step is dropped with the same finality as a token deep in the tail, regardless of how close its score was to the cutoff. A long context produces many such verdicts, and the per-step errors compound. The result is *monotone marginal erosion*: every token whose score is only intermittently above the cutoff is silently and permanently lost.

Reservoir sampling is the streaming-algorithms answer to this setting. Weighted reservoir sampling (Vitter, 1985; Efrimidis and Spirakis, 2006) retains each token with probability proportional to its current score, so that a token’s long-run survival across many steps tracks its *average importance across steps* rather than collapsing on the first step where its score happens to land below the cutoff. We believe this is the right candidate of selection primitive for KV cache eviction, but it inherits the calibration of the score it samples against. The direct-attention magnitude used by existing methods is, however, locally myopic: it counts only what the current query attends to, and misses what we call *bridge tokens* - tokens that no single recent query attends to strongly, but that hold together a strongly-connected cluster of mutually-attended tokens across the context, such that removing one would sever the cluster’s internal connections. Bridges are a canonical instance of marginal-rank tokens: deterministic top- K evicts them at every step, even though the clusters they hold together may be highly important. We therefore lift the reservoir’s input weight from this direct-attention score

to a *Nexus score* that surfaces bridges before any selection happens.

We propose **Nexus Sampling**, a training-free KV cache eviction method built around two new mechanisms that together address both failure modes identified above, applied uniformly across prefill and decode. Upstream, *Nexus scoring* passes the direct-attention score through a short iterative walk recurrence that surfaces bridge tokens before any selection happens. Downstream, *weighted reservoir sampling* draws the K retained tokens with inclusion probability proportional to this walk-augmented weight, replacing the deterministic top- K selection step that every prior eviction method shares. Each fix recovers a category of importance the existing template cannot see: the walk recovers bridges that direct-attention top- K misses, and the reservoir recovers the marginal-rank tokens that any deterministic top- K erodes. Theoretically, Nexus’s long-run token survival decays as a product of per-step inclusion probabilities (a mean over steps), where deterministic top- K ’s collapses to zero on the first below-cutoff step (details in App. A). The contributions are:

- We identify a fundamental limitation of deterministic-top- K KV cache eviction: it converts transient marginality into permanent loss, producing *monotone marginal erosion* in the streaming, fixed-budget regime that is inherent to KV cache eviction.
- We propose **Nexus Sampling**, a training-free KV cache eviction method that applies uniformly to prefill and decode, combining weighted reservoir sampling with a *Nexus score* that surfaces *bridge tokens*, tokens anchoring strongly-connected clusters of mutually-attended tokens across the window.
- We establish theoretical guarantees showing that reservoir sampling’s long-run token survival is a *product over steps* where deterministic top- K ’s is a *min over steps*, and empirically demonstrate at 20% density that Nexus Sampling stays within ~ 1 point of dense attention on Long-Bench while outperforming top- K baselines on retrieval-heavy long-context tasks, with up to $10\times$ smaller per-sequence cache memory than dense FlashAttention-2.

2 Nexus Sampling

We introduce **Nexus Sampling**, a KV cache eviction method that runs alongside decoding: at every

eviction step t (whenever the budget is hit during the decoding stream), Nexus produces the retention set S_t via two components applied in sequence (Figure 2): Nexus scoring followed by weighted reservoir selection. Nexus scoring passes the per-block attention score through a short iterative walk recurrence that surfaces *bridge tokens*, tokens anchoring strongly-connected clusters of mutually-attended tokens, and yields a per-block sampling weight $\mathbf{w}^{(t)}$ that reflects indirect importance. The reservoir step then draws the K retained blocks with inclusion probability proportional to $\mathbf{w}^{(t)}$ under the weighted-without-replacement law, replacing the deterministic top- K selection that every prior eviction method shares. The name reflects the two roles each component plays at every step in the stream: Nexus scoring identifies the *nexus* blocks that hold a cluster together, and the reservoir *samples* from the resulting weights so that every positive-weight block keeps positive inclusion probability rather than being deterministically condemned the first time it lands at the marginal rank.

2.1 Preliminaries and Notation

Streaming view. We treat KV cache eviction as a streaming problem over the steps of inference. At each step t , the LLM ingests one new query $q_t \in \mathbb{R}^{1 \times D}$ (a prompt token during prefill or a decoded token during decode), attends over the current cache K_t, V_t , and appends one new (key, value) pair, after which the cache may exceed the memory budget. An *eviction step* occurs whenever the budget is hit: a subset $S_t \subseteq \{1, \dots, T_t\}$ of B positions is retained, the rest are dropped, and inference continues with K_{S_t}, V_{S_t} . Eviction is therefore relevant in both phases: a long prompt can trigger a one-shot eviction at the end of prefill, while a long generation produces a long stream of eviction steps throughout decode. In both cases, eviction is irreversible: positions evicted at step t are not recoverable at any step $t' > t$.

Block granularity. We work at the granularity of *key blocks* rather than individual tokens, amortizing the per-decision cost over b tokens ($b = 32$); blocks are indexed by $j \in \{1, \dots, N_k\}$ with $N_k = \lceil T_k/b \rceil$ where T_k is the number of cached keys.

Per-block base score. Single-query attention is noisy and dominated by the immediate token, so following standard practice (Li et al., 2024; Ghadia et al., 2025) we score blocks against a length- W observation window of recent queries $\mathbf{Q} \in \mathbb{R}^{W \times D}$

on head-averaged representations. The per-block *base score* $\mathbf{a} \in \mathbb{R}^{N_k}$ is the row average of $\hat{\mathbf{S}} = \text{rownorm}(\text{BlockSum}_b(\text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{D})))$:

$$\mathbf{a} = \frac{1}{W} \sum_{w=1}^W \hat{\mathbf{S}}_{w,\cdot}, \quad \sum_j a_j = 1, \quad (1)$$

Notation. Throughout the rest of the section we use W for the observation-window length, $\hat{\mathbf{S}} \in \mathbb{R}^{W \times N_k}$ for the per-row block distribution, $\mathbf{a} \in \mathbb{R}^{N_k}$ for the base score (Eq. 1), $\mathbf{c} \in \mathbb{R}^{N_k}$ for the walk state, $\mathbf{w} \in \mathbb{R}^{N_k}$ for the combined sampling weight, B for the total retained-block budget, and n for the reservoir averaging count. A full notation table appears in App. A.1.

2.2 Nexus Scoring

Nexus scoring assigns each block a two-term *Nexus score*

$$s_j = a_j + \lambda \tilde{c}_j, \quad (2)$$

that captures both the *direct* importance a_j of block j (how much the recent queries attend to it) and an *indirect* importance $\tilde{c}_j$ that measures how strongly block j acts as a *bridge token*, i.e., how strongly it anchors a cluster of mutually-attended tokens across the window even when no single recent query attends to it strongly. The two terms cover complementary failure modes of a single-query attention score: a_j scores blocks the current query has reason to look at; $\tilde{c}_j$ scores blocks the current query has reason to *remember*, by virtue of their role in the recent context. The mixing weight $\lambda \geq 0$ balances the two terms.

A bridge-type multi-hop question makes the distinction concrete: for “What is the nationality of the director of FILM?”, the director’s name appears in neither the question nor the answer, yet every window query that resolves the chain must route through the block that contains it. No single query places a peak on that block, so a_j leaves it in the mid-band where deterministic top-K cannot separate it from noise, while the walk of Eq. 4 accumulates the moderate mass it draws from many queries and lifts it above the cutoff, at 20% density this raises bridge-block retention from 65.9% to 71.2% (App. B).

Both terms are computed from the same per-query rows of $\hat{\mathbf{S}}$. Denote the rows by $\mathbf{a}^{(q)} \in \mathbb{R}^{N_k}$, so $\mathbf{a}^{(q)} = \hat{\mathbf{S}}_{q,\cdot}$ is the block distribution of window query q . The *direct* importance is the uniform row-average,

$$\mathbf{a} = \frac{1}{W} \sum_{q=1}^W \mathbf{a}^{(q)}, \quad (3)$$

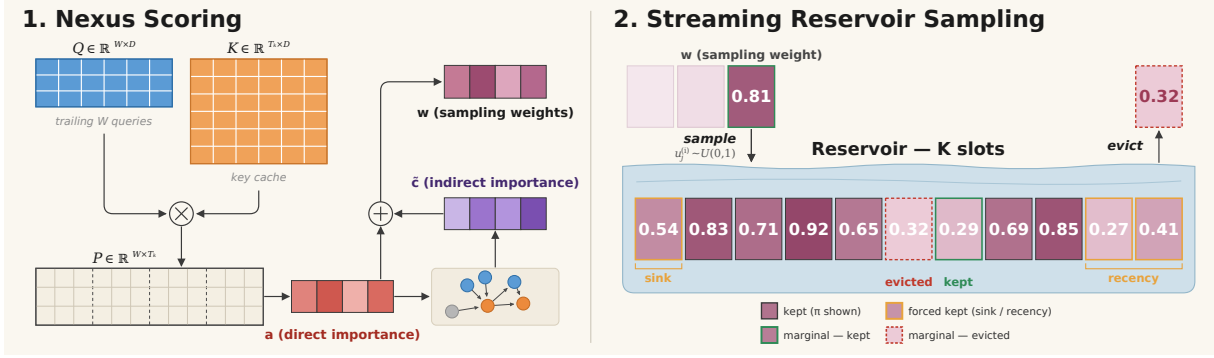


Figure 2: Overview of Nexus Sampling at one eviction step t . *Nexus scoring* (top) combines the direct importance $\mathbf{a}$, the row average of the windowed block attention $\hat{\mathbf{S}}$, with the bridge term $\tilde{\mathbf{c}}$ obtained by an iterative walk over the same query rows, yielding the Nexus score $s_j = a_j + \lambda \tilde{c}_j$ and the sampling weight w_j . *Streaming reservoir sampling* (bottom) consumes $\mathbf{w}$: every block draws a priority π_j and the top $K = B - B_{\text{forced}}$ are retained alongside the forced sink and recency blocks, replacing the deterministic top- K used by every prior eviction method.

which treats each window query as an equal vote. The *bridge* score, by contrast, aggregates the same rows through an alignment-weighted iterative walk. Starting from $\mathbf{C} = \mathbf{0}$, the walk runs for H steps; at each step we plug in a fresh window query row $\mathbf{a}^{(q)}$ and update

$$\mathbf{C} \leftarrow \mathbf{C} + \gamma_q \mathbf{a}^{(q)}, \quad \gamma_q = 1 + \langle \mathbf{a}^{(q)}, \mathbf{C} \rangle, \quad (4)$$

from which $\tilde{\mathbf{c}} = \mathbf{C} / \|\mathbf{C}\|_1$. The per-block weight fed to the reservoir is the Nexus score plus a small recency tie-break,

$$w_j = s_j + \epsilon_{\text{tie}} r_j, \quad r_j = j / (N_k - 1), \quad (5)$$

with r_j a linear ramp from 0 (oldest block) to 1 (newest), and ϵ_{tie} small enough that it does not flip any real signal but large enough to deterministically break ties in favor of more-recent blocks.

2.3 Weighted Reservoir Block Selection

Given the per-block weight $\mathbf{w} \in \mathbb{R}^{N_k}$ from §2.2, the reservoir step decides which K of the N_k blocks survive the eviction step. The choice of selection rule here, not the choice of weight, is what distinguishes Nexus Sampling from every prior eviction method. Following standard practice, we retain the attention-sink block and a small recency window unconditionally; these account for B_{forced} slots of the total budget B , leaving $K = B - B_{\text{forced}}$ slots for the reservoir step below.

n -averaged reservoir priority. For every remaining candidate block j , we draw n independent uniform variates $u_j^{(1)}, \dots, u_j^{(n)} \sim \mathcal{U}(0, 1)$ and form the *n -averaged reservoir priority*

$$\pi_j = \frac{1}{n} \sum_{i=1}^n (u_j^{(i)})^{1/w_j}. \quad (6)$$

We then keep the top K candidates by π_j .

3 Why Nexus Sampling for KV Cache Eviction?

Every existing eviction method makes two implicit choices: that a token’s importance is read off from *direct* per-step attention, and that the K surviving tokens are picked by *deterministic top- K* . Both choices are defensible for a one-shot prefill against a still-dense cache, but both might break in the streaming, fixed-budget regime that defines KV cache eviction (§2). Nexus Sampling replaces them with two components, Nexus scoring and weighted reservoir selection, each addressing a failure mode that the other cannot reach on its own.

Deterministic top- K erases subtly important tokens at the cutoff; the reservoir preserves them.

A streaming eviction policy is applied at every step, and under deterministic top- K a token at rank $K+1$ is dropped with the same finality as a token at rank 10^6 , so a token survives the first e steps only if its weight clears the cutoff at *every* step. Long-run survival is thus a min over steps and collapses to zero on the first below-cutoff step, permanently erasing any token whose per-step weight sits just below the cutoff, even if it is subtly important on average and would be relevant later in the stream. Weighted reservoir sampling (Efrimidis and Spirakis, 2006) replaces this min with a product: per-step inclusion probability is monotone in the weight and strictly positive whenever the weight is, so long-run survival

$$p_j^{(e)} = \prod_{e' \leq e} \Pr(j \in \text{top-}K \text{ by } \pi^{(e')})$$

stays bounded away from zero whenever $w_j^{(e')}$ does (§2.3; Lemma 4.1). The reservoir thus fixes the *selection rule*: it converts transient marginality into a controlled per-step probability rather than an irreversible verdict, and exactly the subtly important tokens the deterministic rule would erase are the ones it keeps at a controlled non-zero rate.

Direct attention misses bridge tokens; Nexus scoring lifts them. *Bridge tokens* are a canonical instance of the subtly important tokens above: tokens that no single recent query attends to *strongly*, but that sit at the intersection of many mutually-attended tokens in the window and structurally hold their cluster together. The direct score $\mathbf{a}$ is a uniform average of per-query distributions and does not reward this structural role: a bridge typically picks up moderate mass from many queries but rarely a peak from any one, so it lands in the noisy mid-band of the heavy-tailed score distribution (Zhang et al., 2023; Le et al., 2026a) rather than in the head. Even a perfect selection rule cannot recover such a token from this score, because sampling (or top- K) proportional to a weight that does not see the structural signal samples from the wrong distribution. The bridge term $\tilde{c}_j$ in the Nexus score (§2.2) is built precisely to extract this higher-order signal: an iterative walk over the same per-query rows compounds mass on tokens that successive queries *agree on*, amplifying exactly the cluster-anchoring positions a uniform average flattens out, which we formalize as *hub amplification* (Theorem 4.2). The walk thus fixes the *weight construction*: it ensures the input to selection already reflects indirect importance, so the reservoir’s preservation guarantee applies to bridges rather than to a wrong notion of importance.

Both fixes are needed, and they compose. The two components are not redundant; each guards against a failure mode the other cannot. The reservoir alone cannot save bridge tokens: if they are not lifted into the head of the weight, sampling proportional to weight samples from the wrong distribution. The walk alone cannot save subtly important tokens that still sit at the cutoff after scoring: a comprehensive per-step weight is still terminated by deterministic top- K the first time it lands below the cutoff. Bridge tokens are recovered upstream in the weight; near-cutoff subtly important tokens are recovered downstream in the selection; together they remove both of the implicit assumptions above. The empirical picture in §5 matches: existing meth-

ods do well when the score has a clear head and the relevant tokens are stably top-ranked, and existing methods degrade on multi-hop retrieval, long summarization, and tasks whose answer depends on subtly important tokens that the recent query does not directly attend to.

4 Theoretical Guarantees

The arguments of §3 have formal counterparts. The reservoir’s long-run survival decays as a *product* over eviction steps where deterministic top- K ’s collapses to zero on the first below-cutoff step (Lemma 4.1), the walk *provably amplifies* hub tokens above the uniform window average (Theorem 4.2), and the two together yield an end-to-end bound on per-step eviction quality (Proposition 4.3). Proofs and supporting lemmas are deferred to App. A.

Lemma 4.1 (Min over steps vs. product over steps). *Consider a sequence of eviction steps $e = 1, \dots, E$ with per-step weights $w_j^{(e)} > 0$ at every step, and let $w_{(K)}^{(e)}$ denote the K -th largest weight at step e . Then the long-run survival probabilities of block j under deterministic top- K and weighted reservoir sampling satisfy*

$$S_j^{\text{top-}K}(E) = \prod_{e=1}^E \mathbf{1}[w_j^{(e)} \geq w_{(K)}^{(e)}],$$

$$S_j^{\text{res}}(E) = \prod_{e=1}^E q_j^{(e)},$$

where $q_j^{(e)} \in (0, 1]$ is strictly positive whenever $w_j^{(e)} > 0$. The first product collapses to zero as soon as any single step has $w_j^{(e)} < w_{(K)}^{(e)}$; the second stays bounded away from zero as long as $w_j^{(e)}$ does.

Lemma 4.1 formalizes the min-vs-product contrast from §3: deterministic top- K converts a single below-cutoff step into an irreversible verdict, while reservoir sampling decays survival only as the product of per-step inclusion probabilities, exactly the property that lets subtly important tokens at the cutoff have a chance to persist across a long stream.

Theorem 4.2 (Hub amplification). *Assume the H walk-step rows $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(H)}$ are i.i.d. with mean $\mathbf{a}_\star$ and covariance Σ , and define the hub score $h_j = \Sigma_j^\top \mathbf{a}_\star$. The walk recurrence (4) satisfies*

$$\mathbb{E}[\tilde{c}_j^{(H)}] = a_{\star,j} + \frac{D_H}{B_H} h_j + O(\|\Sigma\|_F^2),$$

with $B_H \geq H$ and $D_H \geq H(H-1)/2$ both block-independent. In particular, $\mathbb{E}[\tilde{c}_j^{(H)}] \gtrless a_{\star,j} \iff$

$h_j \geq 0$: *hub blocks are amplified above the un-walked score and peripheral blocks are attenuated.*

Theorem 4.2 formalizes the bridge-recovery claim of §3: the walk does not merely denoise the direct score, it injects a block-dependent term proportional to the hub score h_j , which is exactly the higher-order signal a uniform average a over per-query rows cannot recover. Because reservoir inclusion is monotone in the sampling weight (App. A.2), the hub amplification carries through to the selection step: bridge-like tokens that the direct score leaves in the noisy mid-band are lifted into the head of the Nexus score, and the reservoir then keeps them at the controlled non-zero rate of Lemma 4.1. App. B measures this amplification against an annotation-derived bridge set, independent of any attention score. Lemma 4.1 and Theorem 4.2 are component-level guarantees on selection and scoring; we close the loop by combining them into an end-to-end bound on the retained-utility error of one eviction step. Let $z_j \geq 0$ denote the utility that block j would contribute to future attention if it remained in cache (e.g., its future block-attention mass over a short horizon).

Proposition 4.3 (Eviction-quality bound). *Fix one eviction step and let I_j be the reservoir inclusion indicator with $p_j = \Pr(I_j = 1)$. The Horvitz–Thompson estimator $\hat{Z}_{\text{HT}} = \sum_j I_j z_j / p_j$ is unbiased for $Z = \sum_j z_j$, and if $p_j \geq p_{\min} > 0$ and $0 \leq z_j \leq Z_{\max}$, then with probability $\geq 1 - \delta$,*

$$|\hat{Z}_{\text{HT}} - Z| \leq Z_{\max} \sqrt{KN_k / (\delta p_{\min})}.$$

Moreover, if the Nexus weight approximates future utility as $\|\mathbf{w} - \mathbf{z}\|_{\infty} \leq \eta$, then the expected evicted utility $L = \sum_j (1 - I_j) z_j$ satisfies

$$|\mathbb{E}[L] - \sum_j (1 - p_j) w_j| \leq \eta N_k.$$

Proposition 4.3 separates the two sources of approximation: the reservoir step (Lemma 4.1) contributes no systematic bias once inclusion probabilities are accounted for, so the remaining error is controlled by how well the Nexus weight $\mathbf{w}$ tracks the future utility $\mathbf{z}$, which is precisely what the walk (Theorem 4.2) is designed to improve over the direct score $\mathbf{a}$.

5 Experiments

We evaluate Nexus Sampling on two long-context benchmarks under two eviction regimes, comparing against competitive baselines in each regime.

5.1 Setup

Models. Models. We evaluate three long-context instruction-tuned models at different scales: Llama-3.1-8B-Instruct, Llama-3.2-1B-Instruct (Team, 2024), and Qwen3-8B (Team, 2025), all supporting context lengths up to 128K tokens, with the default chat template for each model. **Benchmarks.** We evaluate on two complementary long-context benchmarks: (i) **LongBench** (Bai et al., 2024), spanning single- and multi-document QA, summarization, to few-shot tasks; and (ii) **RULER** (Hsieh et al., 2024), a synthetic diagnostic stressing retrieval and position-sensitive reasoning over very long contexts (4K–64K tokens). On RULER we report average accuracy across the 13-task suite at each context length. **Baselines.** We compare Nexus Sampling against **SnapKV** (Li et al., 2024) and **PyramidKV** (Cai et al., 2025) in the prefill-only setting and against **H2O** (Zhang et al., 2023) and **MorphKV** (Ghadia et al., 2025) in the prefill+decode setting, with **Full Attention** as an upper-reference. *Prefill* blocks evict only during prefill; *Prefill + Decode* blocks evict throughout the entire inference stream. **Bold** marks the leading method per row across the eviction methods (Full Attention shown for reference). We additionally report Ada-KV variants (Ada-SnapKV, Ada-PyramidKV), which layer the head-adaptive budget allocation of Feng et al. (2025) on top of each score; see Appendix C.2 for the isolated comparison. All methods are at 20% density (80% eviction).

Implementation details. We implement Nexus Sampling on top of HuggingFace Transformers, with block-wise eviction at block size $b = 32$ and dense attention on the retained cache served by FlashAttention-2 kernels (Dao, 2024). Observation window $W = 16$; walk depth $H = 3$; mixing weight $\lambda = 0.5$; tie-break magnitude $\varepsilon_{\text{tie}} = 10^{-6}$; reservoir averaging $n = 5$. All experiments run on a single NVIDIA H200 (143 GB) with greedy decoding.

5.2 LongBench Results

Table 1 reports per-task LongBench accuracy at 80% eviction. In the *prefill-only* regime, Nexus Sampling matches or exceeds SnapKV and PyramidKV on the average across all three models, recovering most of the gap to Full Attention. Additional models, densities, and the Ada-KV variants are deferred to App. C.2–C.3. In the *prefill + decode* regime, where eviction errors com-

Table 1: LongBench accuracy at 80% KV cache eviction across three models and 16 tasks.

Model	Method	Density	WIKI	GOV	HPQA	LCC	MNews	MFQA	MUS	NQA	COUNT	Retr.	QAS	QMS	REPO	SamS	TREC	TRIV	Avg.
<i>Prefill-only eviction.</i>																			
Llama-3.1-8B	Full Attention	100%	47.75	34.70	56.96	55.20	26.76	56.16	32.77	29.91	9.66	99.50	45.06	25.39	47.80	43.27	73.00	92.14	48.50
	SnapKV	20%	47.17	28.69	58.61	55.42	23.19	55.44	31.53	30.28	10.25	99.50	40.03	24.60	48.75	42.17	68.50	91.43	47.22
	PyramidKV	20%	47.96	28.07	57.33	53.46	22.37	56.11	32.97	30.49	10.33	99.50	42.36	24.90	47.02	42.53	71.00	91.68	47.38
	Nexus Sampling	20%	48.82	30.07	58.03	54.62	22.65	57.83	32.68	29.24	10.00	99.00	43.47	25.24	47.87	42.93	73.00	92.66	48.01
Llama-3.2-1B	Full Attention	100%	31.23	29.65	35.47	29.83	25.88	43.35	18.45	20.40	3.67	4.50	16.35	21.84	36.04	39.77	61.50	78.54	31.03
	SnapKV	20%	30.45	22.92	35.48	30.75	19.61	39.04	17.15	21.58	3.67	5.00	15.06	21.18	35.86	37.40	58.00	79.31	29.53
	PyramidKV	20%	28.40	20.27	34.89	28.62	16.92	39.91	16.19	19.63	3.00	5.00	14.47	20.80	35.13	37.37	58.00	78.45	28.57
	Nexus Sampling	20%	31.86	24.58	34.52	30.79	20.91	41.87	17.53	20.75	5.67	5.00	15.30	21.74	36.23	37.71	58.50	79.95	30.18
Qwen3-8B	Full Attention	100%	42.62	33.58	57.94	57.39	24.82	52.94	34.33	27.64	4.50	100.00	47.91	23.85	56.67	44.17	71.50	90.71	48.16
	SnapKV	20%	42.38	28.76	56.94	58.07	20.29	52.76	33.92	27.96	6.00	100.00	43.55	23.17	56.60	43.90	66.50	90.71	46.97
	PyramidKV	20%	39.56	26.87	55.25	55.24	18.44	48.48	31.78	26.02	6.50	100.00	36.61	22.29	53.29	43.40	68.00	90.55	45.14
	Nexus Sampling	20%	42.49	31.73	57.68	56.93	22.11	52.52	33.73	27.44	5.00	99.50	43.72	23.89	56.23	43.85	71.00	91.36	47.45
<i>Prefill + decode eviction.</i>																			
Llama-3.1-8B	Full Attention	100%	47.75	34.70	56.96	55.20	26.76	56.16	32.77	29.91	9.66	99.50	45.06	25.39	47.80	43.27	73.00	92.14	48.50
	H2O	20%	48.52	28.05	56.97	52.67	22.17	53.24	32.14	29.87	8.96	99.00	38.78	24.42	45.88	43.37	73.00	91.65	46.79
	MorphKV	20%	48.30	28.41	57.15	54.63	23.17	54.14	30.09	30.35	10.12	99.50	38.96	24.26	47.76	42.51	52.50	92.36	45.89
	Nexus Sampling	20%	48.06	29.44	57.61	53.74	22.15	56.78	32.37	30.89	10.00	99.00	41.16	25.23	46.64	42.75	70.00	92.62	47.40
Llama-3.2-1B	Full Attention	100%	31.23	29.65	35.47	29.83	25.88	43.35	18.45	20.40	3.67	4.50	16.35	21.84	36.04	39.77	61.50	78.54	31.03
	H2O	20%	30.80	23.38	35.45	28.62	21.00	39.66	17.63	20.65	2.67	4.50	14.62	21.16	34.60	38.92	61.50	78.50	29.60
	MorphKV	20%	28.89	23.58	33.64	28.69	21.51	34.95	16.78	19.56	3.17	4.50	14.50	21.17	35.12	36.77	47.00	79.50	28.08
	Nexus Sampling	20%	30.97	24.63	34.41	30.46	20.81	41.16	17.62	21.59	4.64	5.00	15.18	21.01	37.48	38.73	62.00	78.88	30.29
Qwen3-8B	Full Attention	100%	42.62	33.58	57.94	57.39	24.82	52.94	34.33	27.64	4.50	100.00	47.91	23.85	56.67	44.17	71.50	90.71	48.16
	H2O	20%	42.08	29.60	57.83	55.19	19.50	51.76	33.73	28.18	3.50	100.00	43.72	23.28	55.10	44.81	71.50	90.71	46.91
	MorphKV	20%	40.15	29.44	55.96	57.36	20.88	50.66	31.51	27.26	4.50	100.00	40.45	23.01	56.32	43.64	53.50	90.80	45.34
	Nexus Sampling	20%	42.38	30.80	59.21	56.79	20.60	52.17	34.11	26.96	5.00	100.00	43.95	23.31	56.29	43.10	70.00	91.36	47.25

pound across decoding steps and the reservoir’s product-over-steps guarantee (Lemma 4.1) is structurally relevant, Nexus Sampling wins the average against baselines. The gain on MultiFieldQA-en (MFQA) is especially indicative of the marginal-erosion failure mode: in the prefill + decode regime Nexus Sampling tops both deterministic top- K baselines by $\sim 3\%$ and matches Full Attention. MFQA answers often hinge on a span the recent query does not strongly attend to, a subtly important token that deterministic top- K drops once its score dips below the cutoff and that reservoir sampling instead keeps alive at a controlled rate.

5.3 RULER Across Context Lengths

Table 2 reports RULER accuracy across 4K–64K context lengths (Qwen3-8B caps at 32K, so its average spans 4K–32K). Nexus Sampling leads in nearly every cell, with margins that grow with context length. In *prefill-only* it stays within ~ 3 points of Full Attention on Llama-3.1-8B while SnapKV loses 15–25; on the smaller Llama-3.2-1B its margin over SnapKV nearly doubles. In *prefill + decode* the pattern amplifies: H2O and MorphKV sit ~ 20 points below Nexus at every length, MorphKV degrading further as the stream lengthens, while Nexus stays within ~ 1.5 points of Full Attention on Llama-3.1-8B at 64K and again nearly doubles the next-best baseline, H2O, on Llama-3.2-1B. This is the empirical signature of the min-vs-product survival argument: on retrieval-heavy synthetic tasks where every step’s decisions matter,

Table 2: RULER accuracy (%) across 4K–64K context lengths at 20% KV cache density. Qwen3-8B supports a maximum context length of 32K, so its 64K column cannot be evaluated and is reported as “–”.

Model	Method	Density	4K	8K	16K	32K	64K	Avg.
<i>Prefill-only eviction.</i>								
Llama-3.1-8B	Full Attention	100%	96.15	95.91	95.43	91.35	86.30	93.03
	SnapKV	20%	82.69	68.51	73.8	64.42	73.32	72.55
	PyramidKV	20%	75.00	83.65	86.30	86.30	81.97	82.64
	Nexus Sampling	20%	89.42	88.46	94.23	88.46	89.18	89.95
Llama-3.2-1B	Full Attention	100%	76.68	70.67	65.38	64.66	60.34	67.55
	SnapKV	20%	45.43	42.55	37.02	37.02	30.05	38.41
	PyramidKV	20%	41.11	45.91	43.51	46.88	41.59	43.80
	Nexus Sampling	20%	58.89	61.78	56.97	63.70	61.54	60.58
Qwen3-8B	Full Attention	100%	97.60	95.43	88.46	92.79	–	93.57
	SnapKV	20%	70.49	67.01	73.61	79.17	–	72.57
	PyramidKV	20%	61.54	61.06	65.14	65.38	–	63.28
	Nexus Sampling	20%	77.16	83.41	80.29	82.45	–	80.83
<i>Prefill + decode eviction.</i>								
Llama-3.1-8B	Full Attention	100%	96.15	95.91	95.43	91.35	86.30	93.03
	H2O	20%	68.03	68.99	77.40	74.52	69.23	71.63
	MorphKV	20%	84.62	65.38	66.83	61.30	68.27	69.28
	Nexus Sampling	20%	91.35	91.35	94.47	89.18	84.86	90.24
Llama-3.2-1B	Full Attention	100%	76.68	70.67	65.38	64.66	60.34	67.55
	H2O	20%	36.06	33.65	35.34	40.15	37.26	36.49
	MorphKV	20%	22.84	17.31	15.39	18.75	12.98	17.45
	Nexus Sampling	20%	58.89	59.13	59.14	62.98	58.89	59.81
Qwen3-8B	Full Attention	100%	97.60	95.43	88.46	92.79	–	93.57
	H2O	20%	66.83	71.88	74.04	70.19	–	70.74
	MorphKV	20%	67.31	62.50	61.30	57.45	–	62.14
	Nexus Sampling	20%	73.56	81.49	77.16	84.14	–	79.09

deterministic top- K silently erodes the relevant tokens while reservoir sampling does not. The trend continues at 128K, where Nexus leads MorphKV by 9.86 points in prefill + decode (App. C.1).

5.4 Agentic Coding

In table 3, we further evaluate Nexus Sampling as a multi-turn coding agent over a long, growing interaction trace, scoring *Resolved/Pass@1* on 50

tasks sampled from SWE-bench (Jimenez et al., 2024) with DeepSWE-Preview (Luo et al., 2025), trained on top of Qwen3-32B (Team, 2025) within the R2E-Gym environment (Jain et al., 2025). Two axes vary independently. The *scoring scope* is either *Prefill-only*, compressing the cache once at prefill, or *Prefill + Decode*, evicting throughout the entire generation stream. The *eviction protocol* matters more. *Full context every turn* re-prefills the history and re-compresses from scratch at each agent turn, so the cache is only an inference-time approximation of full attention: evicted tokens can re-enter, and every eviction is conditioned on the current turn’s query, making this a *single-turn* problem of what is relevant *right now*, with full access to the history each time. *True eviction* carries the cache across turns and discards permanently, a genuinely *multi-turn* problem in which the method must commit to keeping or discarding a token *before* knowing which later turn will need it, so mistakes are irreversible and compound over the agent’s lifetime.

Under *Full context every turn*, where evicted tokens can return, Nexus Sampling leads all eviction baselines in both scopes. The shift to *True eviction* is what sets them apart. The PyramidKV family collapses to 1 once evictions are permanent, the irreversible-verdict failure mode of deterministic top- K , while Nexus holds at 8 in both protocols, matching the best baseline in prefill-only and staying competitive MorphKV by one task in prefill + decode.

5.5 Ablations

Nexus Sampling is robust to its hyperparameters: across walk depth H , mixing weight λ , observation window W , block size b , and density (Figure 3), the defaults of §5.1 sit inside a flat region on Llama-3.1-8B across three multi-hop QA tasks (HotpotQA, 2WikiMQA, MultiFieldQA-en). First, the walk-depth sweep saturates by $H = 3$ and is indistinguishable from $H = 5$, matching the rank-1 propagation analysis of §4: a few walk steps are enough to surface bridge tokens, and additional iterations only sharpen what is already there, so H does not need per-task tuning. Second, accuracy is essentially flat from 20% to 80% density on all three tasks, confirming the aggressive setting we report in §5.2–5.3 loses no accuracy headroom; this is the empirical signature of the marginal-mass argument of §3, where reservoir sampling preserves subtly important tokens that

Table 3: Agentic coding accuracy (*Resolved/Pass@1* over 50 tasks) at 20% KV density. *Full context every turn* re-prefills and re-compresses the entire history at every turn (evicted tokens can return); *True eviction* carries a persistent cache across turns (evicted tokens are gone for good). **Bold** marks the leading eviction method per block.

Scope	Protocol	Method	KV density	Resolved/Pass@1
Prefill-only	Full context every turn	Full Attention	100%	12 (24%)
		SnapKV	20%	8 (16%)
		PyramidKV	20%	7 (14%)
		AdaSnapKV	20%	6 (12%)
		AdaPyramidKV	20%	6 (12%)
		Nexus Sampling	20%	9 (18%)
	True eviction	Full Attention	100%	12 (24%)
		SnapKV	20%	8 (16%)
		PyramidKV	20%	1 (2%)
		AdaSnapKV	20%	8 (16%)
		AdaPyramidKV	20%	1 (2%)
		Nexus Sampling	20%	8 (16%)
Prefill + Decode	Full context every turn	Full Attention	100%	12 (24%)
		H2O	20%	1 (2%)
		MorphKV	20%	6 (12%)
		Nexus Sampling	20%	7 (14%)
	True eviction	Full Attention	100%	12 (24%)
		H2O	20%	4 (8%)
		MorphKV	20%	9 (18%)
		Nexus Sampling	20%	8 (16%)

deterministic top- K would erode rather than collapsing on the first below-cutoff step. A detailed discussion appears in App. D.

5.6 Inference Efficiency

We measure the two efficiency axes that long-context eviction must improve under a fixed memory budget: decode throughput and steady-state per-sequence memory. All runs use Llama-3.1-8B-Instruct in bf16 on a single H200 (143 GB), prefill length $T_p = 8K$, batch size 1, against dense FlashAttention-2 (Dao, 2024) and the baselines from Tables 1–2.

Table 4: Decode throughput (tokens/s) and per-step latency (ms) at $T_p = 8K$ prefill, batch size 1, on Llama-3.1-8B-Instruct (H200, bf16). Higher is better for throughput; lower is better for per-step latency. **Bold** marks the best value per row.

Decode D	Metric	Dense	SnapKV	MorphKV	Nexus Sampling
4K	Throughput (tok/s, $\uparrow$)	35.10	40.20	31.40	45.50
	Per-step (ms, $\downarrow$)	28.53	24.85	31.84	22.00
8K	Throughput (tok/s, $\uparrow$)	35.00	34.20	29.20	45.20
	Per-step (ms, $\downarrow$)	28.55	29.27	34.27	22.12
16K	Throughput (tok/s, $\uparrow$)	36.00	35.00	32.80	43.80
	Per-step (ms, $\downarrow$)	27.81	28.55	30.49	22.83

Decode throughput. Table 4 reports decode-time throughput and per-step latency across decode lengths $D \in \{4K, 8K, 16K\}$. Nexus Sampling delivers $1.22\text{--}1.30\times$ throughput and 15–23% lower

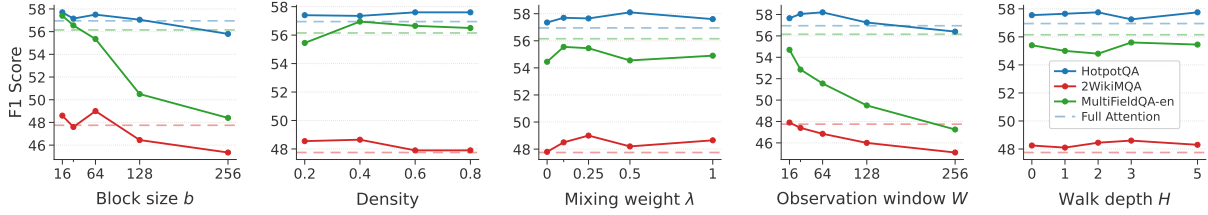


Figure 3: Ablations of the five Nexus knobs on Llama-3.1-8B at 20% density, evaluated on three multi-hop QA tasks (HotpotQA, 2WikiMQA, MultiFieldQA-en).

per-step latency over dense FlashAttention-2, beating both SnapKV and MorphKV at every D .

Table 5: Steady-state per-sequence decode memory on Llama-3.1-8B-Instruct at $T_p = 8K$. Nexus Sampling pins the cache at a constant ≈ 1638 -token budget, so its decode memory is essentially flat in D , while dense FlashAttention-2 grows linearly. **Bold** marks the lower memory footprint per row.

Decode D	Dense FA-2	Nexus Sampling	Saving
4K	1.5 GiB	0.3 GiB	5.0×
8K	2.0 GiB	0.3 GiB	6.7×
16K	3.0 GiB	0.4 GiB	7.5×
32K	5.0 GiB	0.5 GiB	10.0×

Per-sequence memory. Table 5 reports the steady-state per-sequence decode memory (GPU memory each sequence holds during decoding excluding model weights, which determines how many concurrent sequences fit under a fixed GPU budget). Nexus Sampling saves 5.0–10.0× over dense FlashAttention-2, with the saving growing with D because the dense cache scales as $T_p + D$ while Nexus pins the working cache at a constant 20% density of T_p ; at $D = 32K$ this 10× reduction translates into a comparable factor in achievable batch size on the same GPU.

6 Related Work

We situate Nexus Sampling within the broader landscape of KV cache compression and reservoir sampling. KV cache eviction methods (StreamingLLM (Xiao et al., 2024), H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2025), Ada-KV (Feng et al., 2025), and MorphKV (Ghadia et al., 2025)) all reduce to different scoring strategies fed into the same deterministic top- K selection rule, leaving them vulnerable to the irreversible-verdict failure mode discussed in §3. Nexus Sampling is orthogonal to KV cache *selection* (sparse attention) methods such as Quest (Tang et al., 2024),

BLASST (Yuan et al., 2026), and Sketch-and-Walk (Le et al., 2026a), and can be composed with them: eviction decides which tokens stay resident in the cache, and sparse attention then decides which of the surviving tokens to read at each step, so the two stack naturally as a memory-side and a compute-side compression on the same cache. Nexus Sampling also connects to classical weighted reservoir sampling (Chao, 1982; Vitter, 1985; Efraimidis and Spirakis, 2006). See Appendix G for a detailed discussion of each method and its relationship to our approach.

7 Conclusion

We presented **Nexus Sampling**, a training-free KV cache eviction method that pairs an iterative walk surfacing bridge tokens with weighted reservoir sampling, replacing the deterministic top- K selection that every prior eviction method shares. Theoretically, the walk provably amplifies hub tokens above any uniform window average, and the reservoir’s long-run survival decays as a product over eviction steps rather than collapsing on the first below-cutoff step. Empirically, at 80% KV cache eviction, Nexus Sampling matches dense attention within ~ 1 point on LongBench while outperforming top- K baselines on retrieval-heavy long-context tasks, with up to 10× smaller per-sequence cache memory than dense FlashAttention-2.

Implications for agentic context. As LLM workloads shift from one-shot prompts toward long-running agents, eviction moves from a one-time cost to a continuous, lifetime-shaping decision. We expect that in this regime the algorithmic question of *what selection primitive to use* eclipses the question of *what scoring function to use*: existing methods already produce reasonable per-token weights, but only a primitive that preserves marginal mass across a long stream of eviction steps, like the reservoir step at the core of Nexus Sampling, is structurally matched to a streaming-budget setting.

Limitations

Three caveats are worth flagging. First, our theoretical guarantees assume i.i.d. query rows within the observation window (Assumption A.5); in practice consecutive tokens are correlated, and the window acts as a local mixing device rather than producing literally independent samples, so the lemma should be read as characterizing the *tendency* of the walk rather than as an exact finite-sample guarantee. Second, the $n = 1$ configuration is unbiased but per-seed noisy; for deployments where output reproducibility matters more than long-run cache fidelity, the deterministic $n \rightarrow \infty$ branch is the safer default. Third, our streaming-regime gains are largest when the cache experiences *many* update events; for single-prompt workloads the gap to deterministic top- K is correspondingly smaller.

References

- Anthropic. 2025. Claude Sonnet 4.5. Anthropic model release.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multi-task benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling (COLM)*.
- M.-T. Chao. 1982. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656.
- Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- DeepSeek-AI. 2025. [DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning](#). Preprint, arXiv:2501.12948.
- Pavlos S. Efrimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2025. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Gemma Team. 2024. [Gemma: Open models based on Gemini research and technology](#). Preprint, arXiv:2403.08295.
- Ravi Ghadia, Avinash Kumar, Gaurav Jain, Prashant Nair, and Poulami Das. 2025. Dialogue without limits: Constant-sized KV caches for extended responses in LLMs. In *International Conference on Machine Learning (ICML)*.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. [RULER: What’s the real context size of your long-context language models?](#) In *First Conference on Language Modeling*.
- Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. 2025. R2e-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.
- Sahil Joshi, Agniva Chowdhury, Wyatt Bellinger, Amar Kanakamedala, Ekam Singh, Hoang Anh Duy Le, Aditya Desai, and Anshumali Shrivastava. 2026. SOCKET: SOft collision kernel Estimator for sparse attention. *arXiv preprint arXiv:2602.06283*.
- Hoang Anh Duy Le, Sahil Joshi, Zeyu Yang, Zhaozhuo Xu, and Anshumali Shrivastava. 2026a. Scout before you attend: Sketch-and-walk sparse attention for efficient LLM inference. *arXiv preprint arXiv:2602.07397*.
- Hoang Anh Duy Le, Shaochen Zhong, Yifan Lu, Ying-tong Dou, Jiayi Yuan, Yu-Neng Chuang, Xiran Fan, Guanchu Wang, Yuzhong Chen, and Xia Hu. 2026b. FAFO: Lossy KV cache compression for lossless inference acceleration via draftless fumble decoding. In *International Conference on Machine Learning (ICML)*.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpav Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun,

Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. 2025. DeepSWE: Training a fully open-sourced, state-of-the-art coding agent by scaling rl. Notion Blog. Agentica and Together AI.

Mistral AI. 2024. Un Ministral, des Ministraux. Mistral AI model release. <https://mistral.ai/news/ministraux/>.

OpenAI. 2024. OpenAI o1 system card. Technical report, OpenAI.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning (ICML)*.

Llama Team. 2024. *The llama 3 herd of models*. CoRR, abs/2407.21783.

Qwen Team. 2025. *Qwen3 technical report*. CoRR, abs/2505.09388.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57.

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

Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhao Xu, Zirui Liu, and Xia Hu. 2024. KV cache compression, but what must we give in return? A comprehensive benchmark of long context capable approaches. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4623–4648.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Markus Hoehnerbach, Richard Cai, Julien Demouth, John D. Owens, Xia Hu, and 3 others. 2026. *Blasst: Dynamic blocked attention sparsity via softmax thresholding*. In *Proceedings of Machine Learning and Systems*, volume 8, pages 843–859. MLSys.

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. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Appendix

A Theoretical Analysis

This appendix gives full proofs for the three statements of §4 (Lemma 4.1, Theorem 4.2, and Proposition 4.3), together with several supporting statements that we develop only here: reservoir unbiasedness (Lemma A.1), n -averaged concentration (Lemma A.2), window noise reduction (Lemma A.3), and the walk decomposition that underlies Theorem 4.2 (Lemma A.4). Main-text statements are restated verbatim (or in a slightly expanded form when convenient for the proof); appendix-only statements are presented in full.

A.1 Notation

Symbol	Meaning
W	observation window length (recent query tokens)
$\mathbf{Q} \in \mathbb{R}^{W \times D}$	trailing W query rows, head-averaged
$\mathbf{K} \in \mathbb{R}^{T_k \times D}$	full key cache, head-averaged
b	tokens per key block
$N_k = \lceil T_k/b \rceil$	number of key blocks, indexed by j
$\mathbf{P} \in \mathbb{R}^{W \times T_k}$	token-level attention matrix over the observation window
$\mathbf{M} \in \mathbb{R}^{W \times N_k}$	block-aggregated attention;
$\hat{\mathbf{S}} \in \mathbb{R}^{W \times N_k}$	$M_{w,j} = \sum_{t=(j-1)b+1}^{jb} P_{w,t}$ row-normalized block attention;
$\mathbf{a}^{(q)} = \hat{\mathbf{S}}_{q,\cdot}$	$\hat{S}_{w,j} = M_{w,j} / \sum_{j'} M_{w,j'}$ attention distribution of query row q
$\mathbf{a}_* = \mathbb{E}[\mathbf{a}^{(q)}]$	population mean block-attention distribution
Σ	covariance matrix; $\Sigma_{jk} = \text{Cov}(\hat{S}_{q,j}, \hat{S}_{q,k})$
$\mathbf{C} \in \mathbb{R}^{N_k}$	hub score accumulator from the multi-hop walk
$\gamma_q = 1 + \langle \mathbf{a}^{(q)}, \mathbf{C} \rangle$	per-step amplification factor
$h_j = \Sigma_j^\top \mathbf{a}_*$	hub score of block j ; Σ_j is the j -th column of Σ
w_j	combined sampling weight (Eq. 5)
π_j	reservoir priority of block j (Eq. 6)
n	reservoir averaging count
K	block retention budget after forced blocks are removed
e	cache-update event index in the streaming analysis

Table 6: Notation used in §2 and Appendix A.

A.2 Reservoir Sampling: Unbiasedness and Concentration

Lemma A.1 (Reservoir unbiasedness, $n = 1$; Efraimidis and Spirakis, 2006). *Let $\pi_j = u_j^{1/w_j}$ with $u_j \sim \mathcal{U}(0, 1)$ independent, and let $I_j =$*

$\mathbf{1}\{j \in \mathcal{S}\}$ indicate whether block j belongs to the top- K priority set. The selected set $\mathcal{S}$ is a probability-proportional-to-size sample without replacement. In particular $p_j = \Pr(I_j = 1) > 0$ whenever $w_j > 0$, p_j is monotone in w_j , and for any deterministic utility vector $\mathbf{z}$,

$$\widehat{Z}_{\text{HT}} = \sum_{j=1}^{N_k} \frac{I_j z_j}{p_j} \quad \text{satisfies} \quad \mathbb{E}[\widehat{Z}_{\text{HT}}] = \sum_{j=1}^{N_k} z_j.$$

Proof. The priority rule of Efraimidis and Spirakis (2006) is equivalent to drawing independent exponential clocks $E_j = -\log(u_j)/w_j$ with rates w_j and taking the first K arrivals. This is the standard probability-proportional-to-size without-replacement law. Positivity follows because a positive-rate exponential clock can arrive among the first K with non-zero probability; monotonicity follows by coupling two clocks with the same u_j , since u_j^{1/w_j} increases with w_j (equivalently, $-\log(u_j)/w_j$ decreases with w_j). The Horvitz–Thompson identity is immediate: $\mathbb{E}[I_j z_j/p_j] = z_j$ for each j , and summing over blocks gives the claim. $\square$

The statement above is intentionally phrased in terms of inclusion probabilities p_j rather than the simplified formula $K w_j / \sum_j w_j$. For fixed-size weighted sampling without replacement, the exact marginal p_j has no universal linear closed form for all weights and budgets; what Nexus needs is the weaker but correct fact that positive-weight blocks have positive, monotone inclusion probability and admit unbiased retained-mass estimation.

Lemma A.2 (*n*-averaged concentration). *Let $\pi_j^{(n)} = \frac{1}{n} \sum_{i=1}^n (u_j^{(i)})^{1/w_j}$ with $u_j^{(i)} \sim \mathcal{U}(0, 1)$ independent. Then $\mathbb{E}[\pi_j^{(n)}] = w_j/(w_j + 1)$ and $\text{Var}(\pi_j^{(n)}) \rightarrow 0$ at rate $1/n$ as $n \rightarrow \infty$. If the weights w_j are distinct (as guaranteed in practice by the recency tie-break $\varepsilon_{\text{tie}} r_j$), the rank of block j under $\pi_j^{(n)}$ converges almost surely to the rank under $w_j/(w_j + 1)$ as $n \rightarrow \infty$.*

Proof. For $u \sim \mathcal{U}(0, 1)$, the random variable $X = u^{1/w}$ has CDF $F_X(x) = \Pr(u \leq x^w) = x^w$ on $[0, 1]$, density $f_X(x) = w x^{w-1}$, and moments $\mathbb{E}[X^k] = w/(w + k)$. Hence $\mathbb{E}[X] = w/(w + 1)$ and $\text{Var}(X) = w/(w + 2) - (w/(w + 1))^2 = w/[(w + 1)^2(w + 2)]$. Averaging n i.i.d. copies gives $\mathbb{E}[\pi_j^{(n)}] = w_j/(w_j + 1)$ and $\text{Var}(\pi_j^{(n)}) = w_j/[n(w_j + 1)^2(w_j + 2)] = O(1/n)$. By the strong

law of large numbers, $\pi_j^{(n)} \xrightarrow{\text{a.s.}} w_j/(w_j + 1)$, and the monotonicity of $w/(w + 1)$ in w ensures that the rank under $\pi_j^{(n)}$ converges a.s. to the rank under $w_j/(w_j + 1)$. $\square$

Because $w_j/(w_j + 1)$ is strictly increasing in w_j , the deterministic limit induces the same ordering as top- K by w_j directly. The averaging count n is therefore a *single-knob* interpolation between unbiased sampling (Lemma A.1) and deterministic top- K ($n \rightarrow \infty$), with no other change to the pipeline.

A.3 Streaming Marginal-Mass Survival

Lemma 4.1 (Min over steps vs. mean over steps; restated). *Consider a sequence of eviction steps $e = 1, \dots, E$ with per-step weights $w_j^{(e)} > 0$ and a fixed retention budget K at every step. Let $S_j^{\text{top-}K}(E)$ and $S_j^{\text{res}}(E)$ denote the long-run survival probability of block j under deterministic top- K and reservoir sampling respectively. Then $S_j^{\text{top-}K}(E) = \prod_{e=1}^E \mathbf{1}\{w_j^{(e)} \geq w_{(K)}^{(e)}\}$, which collapses to zero as soon as any single step has $w_j^{(e)} < w_{(K)}^{(e)}$; while $S_j^{\text{res}}(E) = \prod_{e=1}^E q_j^{(e)}$ with $q_j^{(e)} \in (0, 1]$ strictly positive whenever $w_j^{(e)} > 0$.*

Proof. Under deterministic top- K , block j survives step e iff $w_j^{(e)} \geq w_{(K)}^{(e)}$. Survival of all E steps is therefore the indicator $\prod_{e=1}^E \mathbf{1}\{w_j^{(e)} \geq w_{(K)}^{(e)}\}$, which is zero whenever a single step falls below the cutoff. Under reservoir sampling, Lemma A.1 gives a positive per-step inclusion probability $q_j^{(e)}$ whenever $w_j^{(e)} > 0$. With independent priority draws at different eviction steps, the survival probability is $\prod_e q_j^{(e)}$, which is strictly positive whenever every $q_j^{(e)}$ is. $\square$

This is the precise sense in which the reservoir step is the *right* primitive for the streaming-budget regime: deterministic top- K converts a single marginal-rank misjudgment into permanent, irrecoverable loss (the block is gone from the cache and cannot re-enter), while reservoir sampling assigns strictly positive inclusion probability at every event, so the long-run survival probability remains bounded away from zero as long as the block’s per-event weight is positive.

A.4 Window Averaging as Noise Reduction

Lemma A.3 (Window noise reduction). *If each row $\mathbf{a}^{(q)} = \boldsymbol{\mu} + \boldsymbol{\xi}_q$ with $\mathbb{E}[\boldsymbol{\xi}_q] = \mathbf{0}$ and per-*

coordinate variance $\leq \nu^2$, then with probability $\geq 1 - \delta$,

$$\|\mathbf{a} - \boldsymbol{\mu}\|_\infty \leq \nu \sqrt{2 \log(2N_k/\delta)/W}.$$

Proof. By construction $\mathbf{a} - \boldsymbol{\mu} = (1/W) \sum_q \boldsymbol{\xi}_q$, an average of W independent zero-mean coordinate-wise sub-Gaussian random vectors. For any block j , Hoeffding's inequality (applied to bounded coordinates of $\boldsymbol{\xi}_q$ on $[-1, 1]$) gives

$$\Pr(|(\mathbf{a} - \boldsymbol{\mu})_j| \geq t) \leq 2 \exp(-Wt^2/(2\nu^2)).$$

Taking a union bound over the N_k block coordinates and solving for t at confidence $1 - \delta$ yields the claim. $\square$

The bound is the standard $1/\sqrt{W}$ noise-reduction rate, with the N_k -dimensional union bound contributing only a $\log N_k$ factor. The implication for the reservoir step is that a modest window length ($W \approx 16$ – 32) suffices to bring the per-block noise floor on the reservoir's input weights below the typical scale of meaningful per-block weight differences.

A.5 Eviction Quality Under Approximate Future Utility

We restate Proposition 4.3 of §4 in its full three-part form (the main-text statement combines parts 1 and 2 into a single bound) and prove it. Let $z_j \geq 0$ denote the utility that block j would contribute to future attention if it remained in cache; this can be instantiated as future block-attention mass, or as a value-norm-weighted utility $z_j = \sum_\tau \alpha_{\tau j} \|V_j\|$ over a short future horizon.

Proposition 4.3 (restated and extended). *Fix one cache-update event and let I_j be the reservoir-sampling inclusion indicator with inclusion probability $p_j = \Pr(I_j = 1)$. For any deterministic future-utility vector $\mathbf{z}$:*

1. *The Horvitz–Thompson retained-utility estimator*

$$\hat{Z}_{\text{HT}} = \sum_{j=1}^{N_k} \frac{I_j z_j}{p_j}$$

is unbiased for the full-cache utility $Z = \sum_j z_j$.

2. *If $p_j \geq p_{\min} > 0$ and $0 \leq z_j \leq Z_{\max}$, then with probability at least $1 - \delta$,*

$$|\hat{Z}_{\text{HT}} - Z| \leq \sqrt{\frac{K}{\delta} \sum_{j=1}^{N_k} \frac{z_j^2}{p_j}} \leq Z_{\max} \sqrt{\frac{KN_k}{\delta p_{\min}}}.$$

3. *If the Nexus weight approximates future utility as $\|\mathbf{w} - \mathbf{z}\|_\infty \leq \eta$, then the expected evicted utility $L = \sum_j (1 - I_j) z_j$ satisfies*

$$\left| \mathbb{E}[L] - \sum_{j=1}^{N_k} (1 - p_j) w_j \right| \leq \eta \sum_{j=1}^{N_k} (1 - p_j) \leq \eta N_k.$$

Proof. The first claim is the Horvitz–Thompson identity already used in Lemma A.1. For the second claim, let $A_j = I_j z_j / p_j$. Because exactly K blocks are retained, $(\sum_j A_j)^2 \leq K \sum_j A_j^2$. Therefore

$$\mathbb{E}[\hat{Z}_{\text{HT}}^2] \leq K \sum_j \mathbb{E} \left[\frac{I_j z_j^2}{p_j^2} \right] = K \sum_j \frac{z_j^2}{p_j}.$$

Since $\text{Var}(\hat{Z}_{\text{HT}}) \leq \mathbb{E}[\hat{Z}_{\text{HT}}^2]$, Chebyshev's inequality yields the stated bound. The final inequality follows from $z_j^2 \leq Z_{\max}^2$ and $p_j \geq p_{\min}$. For the third claim,

$$\mathbb{E}[L] = \sum_j (1 - p_j) z_j,$$

and subtracting $\sum_j (1 - p_j) w_j$ gives $\left| \sum_j (1 - p_j) (z_j - w_j) \right| \leq \eta \sum_j (1 - p_j)$. $\square$

The proposition separates the two sources of approximation. The reservoir step contributes no systematic bias once inclusion probabilities are accounted for; the remaining eviction error is controlled by how well the score $\mathbf{w}$ predicts future utility. This is where the observation window and walk matter: Lemma A.3 reduces estimation noise in direct utility, while Theorem 4.2 raises the weight of hub blocks whose future utility is indirect rather than visible in one-hop attention.

A.6 Multi-Hop Walk: Decomposition and Hub Amplification

This subsection states a rank-1 decomposition of the walk recurrence that underlies the hub-amplification argument, then proves Theorem 4.2.

Lemma A.4 (Walk decomposition and hub-score zero-sum). *Let $\mathbf{M}_q = \mathbf{a}^{(q)} (\mathbf{a}^{(q)})^\top \in \mathbb{R}^{N_k \times N_k}$ be the rank-1 per-query block-affinity matrix. The walk recurrence (4) satisfies the exact identity*

$$\mathbf{C}^{(H)} = \left(\sum_{q=1}^H \mathbf{a}^{(q)} \right) + \left(\sum_{q=1}^H \mathbf{M}_q \mathbf{C}^{(q-1)} \right).$$

Furthermore, $\sum_j h_j = 0$, i.e., the hub scores sum to zero.

Proof. For each step q , the update increment is

$$\begin{aligned}\gamma_q \mathbf{a}^{(q)} &= (1 + \langle \mathbf{a}^{(q)}, \mathbf{C}^{(q-1)} \rangle) \mathbf{a}^{(q)} \\ &= \mathbf{a}^{(q)} + (\mathbf{a}^{(q)\top} \mathbf{C}^{(q-1)}) \mathbf{a}^{(q)} \\ &= \mathbf{a}^{(q)} + \mathbf{M}_q \mathbf{C}^{(q-1)},\end{aligned}$$

an exact equality with no residual. Summing $\mathbf{C}^{(q)} = \mathbf{C}^{(q-1)} + \mathbf{a}^{(q)} + \mathbf{M}_q \mathbf{C}^{(q-1)}$ from $q = 1$ to H with $\mathbf{C}^{(0)} = \mathbf{0}$ yields the stated identity. For the hub-sum claim: since $\sum_j a_j^{(q)} = 1$ a.s., we have $\sum_j \xi_{q,j} = 0$ a.s., so $\sum_j \Sigma_{jk} = \mathbb{E}[\sum_j \xi_{q,j} \xi_{q,k}] = 0$ for every k , giving $\mathbf{1}^\top \Sigma = \mathbf{0}^\top$ and $\sum_j h_j = \mathbf{1}^\top \Sigma \mathbf{a}_* = 0$. $\square$

The first term is the unwalked window aggregate (essentially $W\mathbf{a}$ if all window rows are used). The second term is the genuinely multi-hop contribution: each $\mathbf{M}_q$ is a rank-1 affinity matrix encoding which block pairs query q connects, and $\mathbf{M}_q \mathbf{C}^{(q-1)}$ aggregates the accumulated walk under that affinity structure. The depth H controls the highest-order hop captured; as in Le et al. (2026a), small H (e.g., $H = 3$) suffices in practice because effective composed attention depth in transformer stacks is empirically shallow.

A.6.1 Hub Amplification (Main Theorem)

We first record the i.i.d. stationarity assumption and the formal definition of a hub block, then restate Theorem 4.2 with the full intermediate quantities and prove it.

Assumption A.5 (i.i.d. stationarity). The H walk-step attention distributions $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(H)}$ (i.e., the H query rows consumed by the recurrence (4)) are drawn i.i.d. from a fixed distribution with mean $\mathbf{a}_*$ and covariance matrix Σ .

Definition A.6 (Hub block). Block j is a *hub block* if $h_j = \Sigma_j^\top \mathbf{a}_* > 0$, i.e., queries that give high attention to block j tend to also give high attention to other frequently attended blocks. Block j is *peripheral* if $h_j \leq 0$.

Theorem 4.2 (restated and extended). Under Assumption A.5, the Nexus walk (Eq. 4) with H steps satisfies

$$\mathbb{E}[C_j^{(H)}] = B_H a_{*,j} + D_H h_j + O(H \|\Sigma\|_F^2), \quad (7)$$

where $B_H = \sum_{q=1}^H \mathbb{E}[\gamma_q] \geq H$ is a block-independent amplification factor, $D_H =$

$\sum_{q=1}^{H-1} B_q \geq H(H-1)/2$ is the cumulative covariance-boost coefficient, and the $O(\cdot)$ term captures second-order covariance interactions. Since $\sum_j h_j = 0$ (Lemma A.4), L_1 -normalization gives

$$\mathbb{E}[\tilde{c}_j^{(H)}] = a_{*,j} + \frac{D_H}{B_H} h_j + O(\|\Sigma\|_F^2);$$

in particular,

$$\mathbb{E}[\tilde{c}_j^{(H)}] \geq a_{*,j} \iff h_j \geq 0.$$

Furthermore, hub blocks have strictly higher expected sampling weight $\mathbb{E}[w_j]$; by the monotonicity of inclusion probability in w_j (Lemma A.1), their expected inclusion probability $\mathbb{E}[p_j]$ is strictly higher than the window average $\mathbf{a}$ alone would predict.

Proof. Step 1: Unrolling the recurrence. The walk (Eq. 4) gives $\mathbf{C}^{(H)} = \sum_{q=1}^H \gamma_q \mathbf{a}^{(q)}$, where $\gamma_q = 1 + \langle \mathbf{a}^{(q)}, \mathbf{C}^{(q-1)} \rangle$ and $\gamma_1 = 1$. Taking expectations entry-wise:

$$\begin{aligned}\mathbb{E}[C_j^{(H)}] &= \sum_{q=1}^H \mathbb{E}[\gamma_q a_j^{(q)}] \\ &= \sum_{q=1}^H \left(\mathbb{E}[\gamma_q] a_{*,j} + \text{Cov}(\gamma_q, a_j^{(q)}) \right).\end{aligned} \quad (8)$$

Step 2: Block-independence of the amplification sum. Under Assumption A.5, $\mathbf{a}^{(q)}$ is independent of $\mathbf{C}^{(q-1)}$, so

$$\mathbb{E}[\gamma_q] = 1 + \langle \mathbf{a}_*, \mathbb{E}[\mathbf{C}^{(q-1)}] \rangle.$$

This depends only on $\mathbf{a}_*$ and Σ through $\mathbb{E}[\mathbf{C}^{(q-1)}]$, and crucially *not* on block index j . Defining $B_H = \sum_{q=1}^H \mathbb{E}[\gamma_q]$, the first term in (8) contributes $B_H a_{*,j}$ uniformly across all blocks.

Step 3: Covariance boost. For $q \geq 2$, by the independence of $\mathbf{a}^{(q)}$ from $\mathbf{C}^{(q-1)}$ (Assumption A.5):

$$\begin{aligned}\text{Cov}(\gamma_q, a_j^{(q)}) &= \text{Cov}(\langle \mathbf{a}^{(q)}, \mathbf{C}^{(q-1)} \rangle, a_j^{(q)}) \\ &= \mathbb{E}[\mathbf{C}^{(q-1)}]^\top \Sigma_j,\end{aligned}$$

where Σ_j is the j -th column of $\Sigma = \mathbb{E}[\mathbf{a}^{(q)} \mathbf{a}^{(q)\top}] - \mathbf{a}_* \mathbf{a}_*^\top$. (The equality follows by conditioning on $\mathbf{C}^{(q-1)}$ and using $\mathbb{E}[\mathbf{a}^{(q)} (\mathbf{a}^{(q)})^\top] = \Sigma + \mathbf{a}_* \mathbf{a}_*^\top$.) We now compute $\mathbb{E}[\mathbf{C}^{(q-1)}]$ by induction. The base case is $\mathbb{E}[\mathbf{C}^{(1)}] = \mathbf{a}_* = B_1 \mathbf{a}_*$

since $\gamma_1 = 1$. At each step, $\mathbb{E}[\mathbf{C}^{(q)}] = \mathbb{E}[\mathbf{C}^{(q-1)}] + \mathbb{E}[\gamma_q] \mathbf{a}_* + O(\|\Sigma\|)$, giving

$$\begin{aligned}\mathbb{E}[\mathbf{C}^{(q-1)}] &= B_{q-1} \mathbf{a}_* + O(\|\Sigma\|), \\ B_{q-1} &= \sum_{q'=1}^{q-1} \mathbb{E}[\gamma_{q'}].\end{aligned}$$

Substituting:

$$\text{Cov}(\gamma_q, a_j^{(q)}) = B_{q-1} h_j + O(\|\Sigma\|_F^2).$$

Summing over $q = 2, \dots, H$ and setting $D_H = \sum_{q=1}^{H-1} B_q$:

$$\sum_{q=2}^H \text{Cov}(\gamma_q, a_j^{(q)}) = D_H h_j + O(H\|\Sigma\|_F^2).$$

Combining with Step 2 gives Eq. (7). Note $D_H \geq \sum_{q=1}^{H-1} q = H(H-1)/2$ since $B_q \geq q$.

Step 4: Directional consequence. By Lemma A.4, $\sum_j h_j = 0$. Therefore the L_1 -norm of $\mathbb{E}[\mathbf{C}^{(H)}]$ satisfies

$$\begin{aligned}\sum_j \mathbb{E}[\mathbf{C}_j^{(H)}] &= B_H \underbrace{\sum_j a_{*,j}}_{=1} + D_H \underbrace{\sum_j h_j}_{=0} \\ &\quad + O(H\|\Sigma\|_F^2) \\ &= B_H + O(H\|\Sigma\|_F^2).\end{aligned}$$

Dividing Eq. (7) entry-wise by this norm:

$$\begin{aligned}\mathbb{E}[\tilde{c}_j^{(H)}] &= \frac{B_H a_{*,j} + D_H h_j}{B_H} + O(\|\Sigma\|_F^2) \\ &= a_{*,j} + \frac{D_H}{B_H} h_j + O(\|\Sigma\|_F^2).\end{aligned}$$

Since $D_H/B_H > 0$, we have $\mathbb{E}[\tilde{c}_j^{(H)}] \gtrless a_{*,j}$ if and only if $h_j \gtrless 0$.

Step 5: Hub retention. By Lemma A.1, the inclusion probability $p_j(\mathbf{w}) = \Pr(I_j = 1 \mid \mathbf{w})$ is increasing in w_j with all other weights fixed. The realized weight $w_j = C_j^{(H)}$ is random; taking expectations over $\mathbf{w}$ gives $\mathbb{E}[p_j] = \mathbb{E}_{\mathbf{w}}[p_j(\mathbf{w})]$. Hub blocks satisfy $\mathbb{E}[w_j] = \mathbb{E}[C_j^{(H)}] > a_{*,j}$ after normalization (Step 4), while peripheral blocks have $\mathbb{E}[w_j] < a_{*,j}$. Because $p_j(\mathbf{w})$ is increasing in w_j , a shift in the distribution of w_j toward larger values (higher mean) raises $\mathbb{E}[p_j]$ via the law of total expectation. Consequently, the expected inclusion probability of hub blocks strictly exceeds what the unwalked window score a_j would predict. $\square$

Remark. Assumption A.5 requires i.i.d. query rows, which is an idealization: consecutive tokens within a document are correlated. In practice, the window acts as a local mixing device; the theorem should be read as characterizing the *tendency* of the walk to amplify cross-query consensus blocks rather than as an exact finite-sample guarantee.

B Bridge Tokens: Definition and Evidence

Section 2.2 introduces bridge tokens as blocks that no single window query attends to strongly but that anchor a cluster of mutually-attended tokens, and Theorem 4.2 shows that the walk recurrence amplifies such blocks above the uniform window average. Both are statements about the scoring function rather than about the data. They establish what the walk does to a block that the window reaches only through other blocks it attends to, not whether such blocks occur in real contexts or whether the amplification is large enough to change which blocks an eviction step retains. This appendix supplies the missing evidence. We construct a reference set of bridge blocks that is fixed by dataset annotation rather than by any attention score, and show that the walk term changes whether those blocks survive a fixed-budget eviction step. §B.1 gives the construction, the recall metric and the results at 20% density. §B.2 shows where the reference blocks fall in the two terms of the Nexus score.

B.1 Bridge Block Recall Under Fixed-Budget Eviction

Write M_{ij} for the attention mass that query block i pays to key block j . The blocks at issue are those a decoding query x attends to weakly, M_{xB} small, while B carries the path $A \rightarrow B \rightarrow C$ between blocks that are attended strongly, M_{AB} and M_{BC} large: x never looks at B directly, yet B is its only route to A and C , and evicting it severs that route. A question of the form *In what country was the director of [Film] born?* is the canonical instance. It resolves in two hops, film to director and director to birthplace, and the director's name appears in neither the question nor the answer. Window queries concerning the film and window queries concerning the country both attend to it moderately, and neither attends to it sharply, so its direct score a_j falls in the mid-band and it is a candidate for eviction. Removing it removes the only block joining the two supporting paragraphs, and the answer is no longer recoverable from the retained cache. Identifying such blocks from M would make the measurement circular, since $\tilde{c}_j$ is large by construction on exactly the blocks the walk amplifies.

Reference set. A recall metric is informative only if its reference set is built without consulting the score being measured, so we derive ours from dataset annotation rather than from model internals.

We build $\mathcal{B}$ on three LongBench tasks. On HotpotQA we take the questions annotated as *bridge* type, in which the answer requires an intermediate entity that links the two supporting paragraphs and that does not appear in the question itself, and mark the token positions of that entity. On 2WikiMQA we mark the entity shared by the two evidence triples of the annotated reasoning path, which plays the same linking role. MultiFieldQA-en is single-hop and carries no bridge annotation, so we mark the annotated evidence span, and retention there measures whether the block the answer is drawn from survives rather than whether a linking block does. For each instance we map the marked token positions to key blocks under the block size $b = 32$ of §5.1, giving a reference set $\mathcal{B} \subseteq \{1, \dots, N_k\}$ that is fixed before any method runs and is identical across the methods compared.

Metric. Given a retention set S produced by an eviction method at fixed density, Bridge Block Recall is the fraction of reference blocks that survive eviction,

$$\text{BRR}(S) = \frac{|\mathcal{B} \cap S|}{|\mathcal{B}|}, \quad (9)$$

averaged over instances and then over the three tasks. BRR measures retention rather than task accuracy: it asks whether the blocks anchoring a multi-hop chain are still in the cache after eviction, which separates a scoring failure from a downstream reasoning failure, and it is the quantity Theorem 4.2 speaks to. All numbers below use Llama-3.1-8B-Instruct at a fixed budget of K blocks per head, corresponding to the 20% density of §5.1.

Table 7: Bridge Block Recall (%) at 20% KV cache density with Llama-3.1-8B-Instruct, averaged over HotpotQA, 2WikiMQA and MultiFieldQA-en and measured against the annotation-derived reference sets defined above. The $\lambda = 0$ row ablates the walk term while holding the selection rule, budget, block size and observation window fixed, isolating the contribution of Nexus scoring. **Bold** marks the highest recall.

Method	Density	BRR (%)
H2O	20%	65.3
MorphKV	20%	57.5
Nexus Sampling, $\lambda = 0$ (no walk)	20%	65.9
Nexus Sampling	20%	71.2

Direct-attention scoring drops bridge blocks. With the walk ablated, 34.1% of reference blocks are evicted at 20% density; H2O and MorphKV evict 34.7% and 42.5%. A method can therefore

score well on aggregate long-context benchmarks while discarding a third or more of the blocks that a multi-hop question depends on.

The walk accounts for the difference. The $\lambda = 0$ configuration differs from full Nexus Sampling only in the weight fed to the reservoir, so the 5.3-point gap is attributable to the walk term alone, and is the empirical counterpart of the hub amplification of Theorem 4.2: the blocks the walk lifts out of the mid-band are the ones the annotation independently marks as anchoring the multi-hop chain. The ordering across methods tracks how far each scorer relies on what is attended locally, with the recency-biased MorphKV furthest behind at 57.5. Since BRR measures retention, the link to end-task performance is made by the multi-hop results of §5.2 and the λ ablation of App. D.

B.2 Visualization

Figure 4 plots the two terms of the Nexus score against one another for a single eviction step, with the reference blocks highlighted. The direct score a_j places them in the same low-score band as the bulk of the cache, the failure mode of §2.2, while along $\tilde{c}_j$ they separate from that band, so the combined weight $s_j = a_j + \lambda \tilde{c}_j$ of Eq. 2 ranks them above a cutoff the direct score alone places them below.

C Additional Experiments

C.1 RULER at 128K Context Length

The RULER evaluation of §5.3 stops at 64K, which is where we stopped rather than where the models stop. To confirm that the trend of Table 2 continues into the regime the benchmark was designed to stress, we extend RULER to a 128K context length on Llama-3.1-8B, the largest long-context model in our suite. All settings are unchanged from §5.1: 20% density, block size $b = 32$, observation window $W = 16$, walk depth $H = 3$, mixing weight $\lambda = 0.5$, and reservoir averaging $n = 5$. Table 8 reports the 128K result alongside the 64K column of Table 2, so that the behaviour across the two lengths can be read in one place.

Nexus Sampling remains the leading eviction method at 128K in both regimes. In prefill + decode it leads MorphKV, the strongest baseline at this length, by 9.86 points. In prefill-only it reaches 79.09 against SnapKV’s 67.55.

Both Nexus variants also exceed the Full Attention reference at 128K (79.09 and 80.53 against

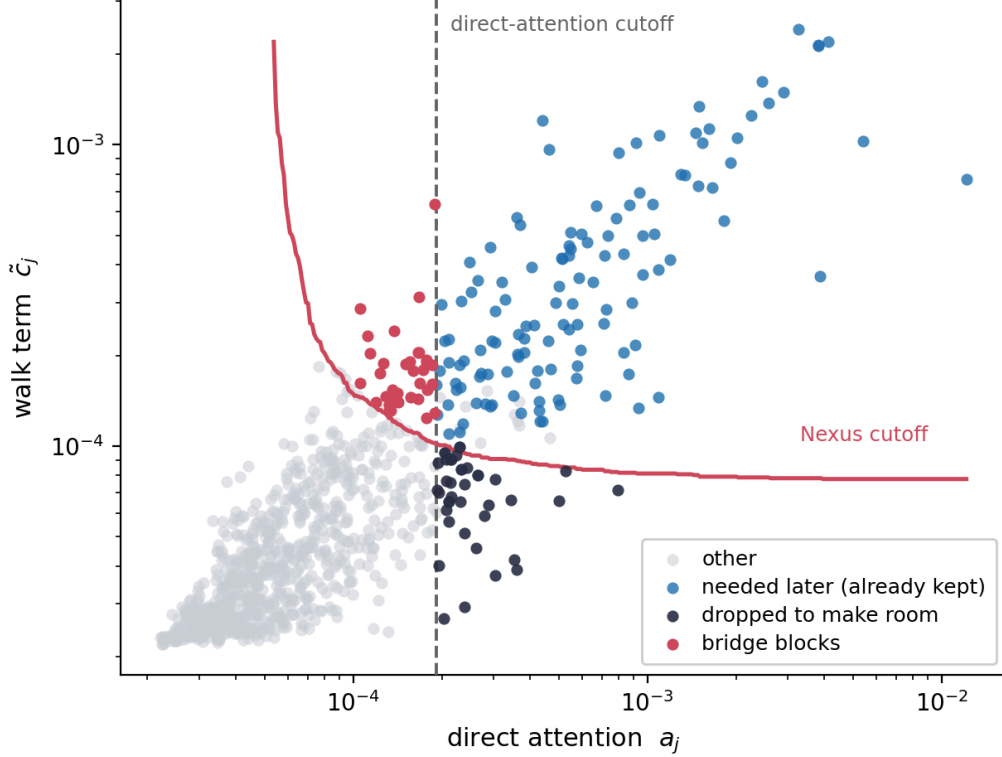


Figure 4: Direct score a_j against bridge score $\tilde{z}_j$ for all key blocks at one eviction step on a bridge-type HotpotQA instance. Oracle bridge blocks (§B.1) are highlighted. Bridges are indistinguishable from the tail along a_j but separate along $\tilde{z}_j$, so the two-term score of Eq. 2 ranks them above the cutoff that the direct score alone would place them below.

Table 8: RULER accuracy (%) on Llama-3.1-8B at 20% density, extending Table 2 to a 128K context length. The 64K column is reproduced from Table 2 so that the trend across the two lengths can be read directly. **Bold** marks the leading eviction method per column within each regime (Full Attention shown for reference).

Method	Density	64K	128K
<i>Prefill-only eviction.</i>			
Full Attention	100%	86.30	78.13
SnapKV	20%	73.32	67.55
Nexus Sampling	20%	89.18	79.09
<i>Prefill + decode eviction.</i>			
Full Attention	100%	86.30	78.13
H2O	20%	69.23	67.31
MorphKV	20%	68.27	70.67
Nexus Sampling	20%	84.86	80.53

78.13). This reads as an anomaly only when 128K is viewed in isolation. The crossover has already occurred at 64K in the prefill-only regime, where Nexus reaches 89.18 against Full Attention’s 86.30 (Table 2), so 128K continues a trend rather than introducing one. The mechanism is visible in the Full Attention row itself: dense attention degrades

once the context exceeds the length over which the model reliably resolves position, losing 8.17 points between 64K and 128K, while Nexus Sampling under prefill + decode eviction loses only 4.33 over the same interval. Eviction at long context is therefore not purely lossy. Discarding low-weight positions also removes distractor mass that dense attention would otherwise integrate over, so a selection rule that retains the right positions can recover accuracy that the dense model loses to long-context dilution. We report this as an observation rather than a general claim about dense attention: these are single-seed numbers (§5.1) on one model at one density.

C.2 Comparison with Head-Adaptive Budget Allocation (Ada-KV)

Ada-KV (Feng et al., 2025) is *orthogonal* to Nexus Sampling: rather than proposing a new token-importance score, it redistributes a fixed layer-wide budget *across attention heads* so that heads with more concentrated attention keep more tokens. It can therefore be layered on top of any scoring rule, including SnapKV (yielding Ada-SnapKV) or, in

principle, Nexus Sampling itself. Because this head-budget allocation is a separate axis from the scoring contribution we study in the main paper, we report Ada-SnapKV separately here rather than mixing it into Table 1.

Table 9 compares Ada-SnapKV against SnapKV and Nexus Sampling in the prefill-only regime, with Full Attention shown for reference. Ada-SnapKV improves over plain SnapKV on average, as expected from its better budget allocation, but Nexus Sampling remains competitive on the average across all three models using a uniform per-head budget, indicating that the scoring contribution we introduce is complementary to the head-adaptive allocation of Ada-KV.

When the same head-adaptive allocation is applied on RULER, the picture is consistent: it lifts every scoring rule, and Nexus Sampling under the Ada-KV allocation (Ada-Nexus Sampling) remains the leading method on average across all three models, confirming that our scoring contribution composes with the orthogonal budget-allocation axis. Table 10 reports the prefill-only RULER results with the Ada-KV allocation; for Qwen3-8B we additionally report results without the chat template, since RULER’s synthetic prompts are sensitive to the template wrapping.

C.3 Additional Models

The main-paper evaluation covers Llama-3.1-8B, Llama-3.2-1B and Qwen3-8B. To check that the gains of Nexus Sampling are not tied to a single pretraining recipe, we repeat the prefill-only LongBench evaluation of Section 5.2 on three further models drawn from different organizations: DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025), Gemma-7B (Gemma Team, 2024) and Ministral-8B-Instruct (Mistral AI, 2024). Every setting is unchanged from the main experiments: 20% density ($B/N_k = 0.20$), the same 16 LongBench tasks, and the same block size and observation-window length. Table 11 reports per-task accuracy. Nexus Sampling is the leading eviction method on average for all three models, and in each case recovers more of the full-attention reference than either SnapKV or PyramidKV, indicating that the scoring contribution we introduce transfers across pretraining recipes rather than exploiting a property of the Llama and Qwen families. Together with the three models of the main paper, this brings our coverage to six models spanning four families.

C.4 Scaling to Llama-3.1-70B

The evaluations of §5 and §C.3 top out at 8B, leaving open whether the gains of Nexus Sampling narrow as capacity grows and the direct-attention score becomes better calibrated on its own. We therefore repeat the prefill-only comparison on Llama-3.1-70B-Instruct at 20% density, holding block size, observation window, walk depth and mixing weight at the values of §5.1.

Nexus Sampling leads both deterministic baselines and comes closest to Full Attention, losing 0.29 points against 0.52 for PyramidKV and 0.70 for SnapKV. We do not read this as a separable gain: all three eviction methods stay within 0.70 points of the dense reference and span only 0.41 points between them. What the comparison does show is that the ordering observed at 1B and 8B does not invert at 70B, and that eviction to 20% density remains close to lossless on a multi-hop task at this scale. A single task and a single seed do not support more.

D Detailed Ablations

This appendix expands §5.5, walking through each panel of Figure 3. All settings match §5.1: Llama-3.1-8B at 20% density, evaluated on three multi-hop QA tasks (HotpotQA, 2WikiMQA, MultiFieldQA-en). We selected this slice deliberately: multi-hop QA stresses the two failure modes Nexus Sampling is designed to address (bridge tokens and irreversible eviction), so any knob that mattered should move the numbers here before showing up on broader benchmarks.

Walk depth H . $H = 0$ (no walk) is already strong because the windowed direct score alone is informative; this is the regime every direct-attention baseline already operates in, which is why those baselines are competitive on retrieval tasks where a single bridge hop suffices. $H \geq 1$ adds a small but consistent improvement on 2WikiMQA, the most multi-hop-sensitive of the three tasks: each additional walk step lets a token inherit influence from a neighbor one further hop away from the current query, and 2WikiMQA is precisely the dataset where the answer-supporting token is reached through such an intermediate. $H \in \{2, 3\}$ captures the bulk of the gain and $H = 5$ is essentially indistinguishable from $H = 3$, matching the rank-1 propagation analysis of App. A: after a few steps the walk converges toward the principal direction of the attention operator and additional

Table 9: Prefill-only LongBench accuracy at 20% density, isolating the orthogonal head-adaptive budget allocation of Ada-KV (Feng et al., 2025). Ada-SnapKV applies the Ada-KV allocation on top of the SnapKV score; Nexus Sampling uses a uniform per-head budget. **Bold** marks the leading method per row across the eviction methods (Full Attention shown for reference).

Model	Method	Density	WIKI	GOV	HPQA	LCC	MNews	MFQA	MUS	NQA	COUNT	Retr.	QAS	QMS	REPO	SamS	TREC	TRIV	Avg.
Llama-3.1-8B	Full Attention	100%	47.75	34.70	56.96	55.20	26.76	56.16	32.77	29.91	9.66	99.50	45.06	25.39	47.80	43.27	73.00	92.14	48.50
	SnapKV	20%	47.17	28.69	58.61	55.42	23.19	55.44	31.53	30.28	10.25	99.50	40.03	24.60	48.75	42.17	68.50	91.43	47.22
	Ada-SnapKV	20%	48.42	29.34	57.90	55.05	22.82	56.32	32.93	30.42	10.08	99.50	43.73	25.15	47.89	42.20	72.00	92.36	47.88
	Nexus Sampling	20%	48.82	30.07	58.03	54.62	22.65	57.83	32.68	29.24	10.00	99.00	43.47	25.24	47.87	42.93	73.00	92.66	48.01
Llama-3.2-1B	Full Attention	100%	31.23	29.65	35.47	29.83	25.88	43.35	18.45	20.40	3.67	4.50	16.35	21.84	36.04	39.77	61.50	78.54	31.03
	SnapKV	20%	30.45	22.92	35.48	30.75	19.61	39.04	17.15	21.58	3.67	5.00	15.06	21.18	35.86	37.40	58.00	79.31	29.53
	Ada-SnapKV	20%	28.58	24.00	35.78	29.75	20.59	40.82	18.24	20.87	2.64	4.50	15.88	21.52	36.25	37.47	61.00	79.15	29.82
	Nexus Sampling	20%	31.86	24.58	34.52	30.79	20.91	41.87	17.53	20.75	5.67	5.00	15.30	21.74	36.23	37.71	58.50	79.95	30.18
Qwen3-8B	Full Attention	100%	42.62	33.58	57.94	57.39	24.82	52.94	34.33	27.64	4.50	100.00	47.91	23.85	56.67	44.17	71.50	90.71	48.16
	SnapKV	20%	42.38	28.76	56.94	58.07	20.29	52.76	33.92	27.96	6.00	100.00	43.55	23.17	56.60	43.90	66.50	90.71	46.97
	Ada-SnapKV	20%	42.25	30.66	57.73	57.36	20.57	52.12	34.28	27.83	5.50	100.00	45.27	23.90	56.95	43.21	70.50	90.71	47.43
	Nexus Sampling	20%	42.49	31.73	57.68	56.93	22.11	52.52	33.73	27.44	5.00	99.50	43.72	23.89	56.23	43.85	71.00	91.36	47.45

Table 10: Prefill-only RULER accuracy (%) across 4K–64K context lengths at 20% density with the orthogonal Ada-KV (Feng et al., 2025) head-adaptive budget allocation applied to each scoring rule. **Bold** marks the leading method per row across the eviction methods (Full Attention shown for reference).

Model	Method	Density	4K	8K	16K	32K	64K	Avg.
Llama-3.1-8B	Full Attention	100%	96.15	95.91	95.43	91.35	86.30	93.03
	Ada-SnapKV	20%	82.21	89.18	94.23	92.79	83.89	88.46
	Ada-PyramidKV	20%	84.86	89.18	91.83	87.26	81.97	87.02
	Ada-Nexus Sampling	20%	85.80	91.30	94.00	88.70	84.60	88.88
Llama-3.2-1B	Full Attention	100%	76.68	70.67	65.38	64.66	60.34	67.55
	Ada-SnapKV	20%	57.69	54.33	50.00	51.68	44.71	51.68
	Ada-PyramidKV	20%	56.97	52.64	43.99	49.52	38.46	48.32
	Ada-Nexus Sampling	20%	57.45	52.88	52.64	56.49	50.72	54.04
Qwen3-8B	Full Attention	100%	97.60	95.43	88.46	92.79	–	93.57
	Ada-SnapKV	20%	73.56	73.80	79.57	77.41	–	76.09
	Ada-PyramidKV	20%	65.38	66.11	69.95	74.28	–	68.93
	Ada-Nexus Sampling	20%	75.48	81.49	83.41	84.38	–	81.19
<i>Qwen3-8B without chat template.</i>								
Qwen3-8B	Full Attention	100%	97.60	95.43	88.46	92.79	–	93.57
	Ada-SnapKV	20%	81.73	89.66	88.70	90.38	–	87.62
	Ada-PyramidKV	20%	81.73	86.78	85.10	88.94	–	85.64

iterations only sharpen what is already there. We therefore default to $H = 3$, which keeps the bridge-token benefit while leaving the per-step cost negligible relative to dense attention.

Mixing weight λ . $\lambda = 0$ closes most of the gap to Nexus but stays consistently below $\lambda > 0$ on 2WikiMQA, confirming that the bridge term contributes signal beyond what direct attention already captures. $\lambda = 1.0$ overweights the bridge term and slightly hurts HotpotQA, where the answer is often a single hop away and the direct score is the cleaner signal; mixing the two terms is what makes the score robust across multi-hop structures of different depth. The safe operating range is $\lambda \in [0.25, 0.5]$ across all three datasets, and within that range the surface is flat, so the choice does not require per-task tuning; we default to $\lambda = 0.5$ throughout.

Observation window W . Small windows ($W \in \{8, 16, 32\}$) track full attention closely. Larger

windows degrade sharply on tasks that need short-range responsiveness: MultiFieldQA-en and 2WikiMQA both drop ~ 5 – 10 points at $W = 256$, because the per-query distribution gets diluted across queries that are no longer relevant to the current decoding step. This is a familiar effect from streaming attention-scoring methods: averaging the score over too long a window turns the per-step decision into a stale running mean and erases the locality the model relies on. $W = 16$ is a sweet spot, small enough to remain responsive yet large enough to absorb single-step noise, and matches the window sizes reported by recent eviction work (Li et al., 2024; Ghadia et al., 2025).

Block size b . $b \in \{16, 32, 64\}$ are within noise of one another, while $b \geq 128$ degrades sharply. The mechanism is straightforward: too-coarse blocks aggregate over too many tokens at once, hiding the single-token signal that the per-block attention distribution needs to resolve. The flat region at small b is what makes block-wise eviction practical at all: it lets implementations align the eviction granularity with the hardware-friendly block sizes already used by FlashAttention kernels without paying an accuracy tax. We default to $b = 32$, which is the smallest block size that lets the kernel achieve full memory-bandwidth utilization on the H200.

Density. Accuracy is essentially flat from 20% to 80% density on all three tasks, confirming that the aggressive 20% setting we report in §5.2–5.3 does not give up accuracy headroom to less aggressive configurations. This is the empirical signature of the marginal-mass argument of §3: under reservoir sampling, even tight budgets retain the subtly important tokens that deterministic top- K would erode, and the long-run survival probability decays gracefully with the budget rather than collapsing on

Table 11: LongBench accuracy at 80% KV cache eviction ($B/N_k = 0.20$, i.e., $Density = 20\%$) across three additional models and 16 tasks. All three methods evict only during the prefill phase. **Bold** marks the leading method per row across the eviction methods (Full Attention shown for reference).

Model	Method	Density	WIKI	GOV	HPQA	LCC	MNews	MFQA	MUS	NQA	COUNT	Retr.	QAS	QMS	REPO	SamS	TREC	TRIV	Avg.
<i>Prefill-only eviction.</i>																			
DeepSeek-R1-Distill-Llama-8B	Full Attention	100%	14.41	29.63	14.82	31.17	23.92	40.30	10.66	14.56	11.50	23.50	24.41	21.80	37.84	41.64	68.50	87.13	30.99
	SnapKV	20%	14.17	26.79	15.45	31.98	19.74	38.74	10.84	14.71	10.00	30.92	22.27	21.93	37.07	41.23	67.00	87.13	30.62
	PyramidKV	20%	14.30	26.52	14.34	29.78	19.53	39.07	10.48	14.34	9.50	28.92	20.94	21.77	36.79	40.35	67.50	87.05	30.07
	Nexus Sampling	20%	14.30	27.32	14.71	32.10	19.55	38.66	10.15	14.90	9.12	31.50	22.24	22.23	37.79	41.37	69.00	87.30	30.77
Gemma-7B	Full Attention	100%	24.19	26.60	27.22	42.21	23.73	44.76	7.13	14.68	1.50	54.50	34.49	19.23	42.63	36.24	72.50	80.58	34.51
	SnapKV	20%	23.25	23.64	27.85	43.83	21.48	44.72	6.94	14.75	1.64	53.50	33.85	19.27	44.58	36.45	69.00	79.59	34.02
	PyramidKV	20%	23.43	21.88	27.97	41.72	19.48	43.90	6.90	14.72	1.15	53.50	32.38	19.05	43.60	35.94	66.50	79.66	33.24
	Nexus Sampling	20%	24.63	22.81	27.84	45.50	20.12	44.67	6.88	14.72	1.00	53.00	33.62	18.75	45.70	35.71	72.00	80.13	34.19
Minstral-8B-Instruct	Full Attention	100%	52.79	32.67	61.40	58.78	26.36	53.15	36.31	26.79	10.00	100.00	47.55	24.33	58.02	46.43	75.50	91.06	50.07
	SnapKV	20%	51.86	28.16	62.05	57.49	22.45	51.36	36.11	26.83	10.50	100.00	45.38	23.82	58.94	45.66	74.50	91.01	49.13
	PyramidKV	20%	51.78	27.79	61.23	57.05	21.71	52.09	35.11	26.43	11.00	100.00	44.81	23.74	58.61	45.01	74.00	90.93	48.83
	Nexus Sampling	20%	52.54	29.16	62.65	57.09	22.84	52.14	35.40	26.30	8.50	100.00	47.26	23.68	57.92	45.29	76.00	90.93	49.23

Table 12: Prefill-only HotpotQA accuracy at 20% KV cache density on Llama-3.1-70B-Instruct. Settings follow §5.1. **Bold** marks the leading eviction method (Full Attention shown for reference).

Method	Density	HotpotQA
Full Attention	100%	66.08
SnapKV	20%	65.38
PyramidKV	20%	65.56
Nexus Sampling	20%	65.79

the first below-cutoff step. Practically, this means the relevant question at deployment time is not “how much can we shrink the cache before accuracy breaks?” but “how much memory headroom does the workload need for other purposes?”, which is the regime modern agentic deployments actually operate in.

Selection rule: weighted reservoir vs. deterministic top- K . Holding the score fixed at the Nexus score s_j and replacing weighted reservoir sampling with deterministic top- K isolates the selection rule on its own, complementing the λ panel above, which isolates the score with the selection rule held fixed. Table 13 reports the swap. The Δ row is strictly non-negative and reaches +9.4 at 16K, so the reservoir contributes a gain that is orthogonal to the score rather than absorbed into it: with the score held fixed, replacing top- K with the reservoir is pure upside. The gain is largest in the middle of the range, which is what the marginal-mass argument of §3 predicts: at 4K and 8K few eviction steps have accumulated and both rules retain nearly everything that matters, while at 32K both rules are budget-bound enough that the task saturates; in between, top- K ’s min-over-steps survival has had time to zero out tokens that the reservoir’s product-over-steps survival still holds. Taken with the λ panel, the two ablations establish that

Table 13: Selection rule ablated in isolation on the CWE task of RULER (Qwen3-8B, 20% density), with the score held fixed at the Nexus score s_j . The only difference between the two rows is how survivors are chosen from that score. **Bold** marks the leading rule per column.

Selection (score = s_j)	4K	8K	16K	32K	Avg.
Deterministic top- K	90.6	96.9	40.6	50.0	69.5
Weighted reservoir (ours)	96.9	100.0	50.0	50.0	74.2
Δ (reservoir – top- K)	+6.3	+3.1	+9.4	0.0	+4.7

each component contributes independently of the other, which is the composition claim of §3 shown empirically.

Reservoir-averaging count n . n controls how many reservoir draws are averaged before survivors are committed at each eviction step, and it interpolates between the two rows of Table 13: small n keeps the selection stochastic, and as n grows the averaged keys concentrate on their expectations, so the rule approaches the deterministic top- K limit. Accuracy is stable across the range, from 73.96 at $n = 20$ to 77.61 at $n = 5$, with no cliff at either extreme; the knob is therefore not one that requires tuning to deploy. The seed variance is the more informative column: it is minimised at $n = 5$ and grows at both ends, for different reasons. At $n = 1$ a single draw carries the full sampling noise of the reservoir, and at $n = 20$ the averaging has removed enough of the stochasticity that the rule inherits the brittleness of deterministic selection, which shows up as sensitivity to which tokens happened to fall below the cutoff early in the stream. We default to $n = 5$, which attains both the best average and the lowest variance, and which keeps the per-step cost of the averaging negligible relative to the walk.

Table 14: Reservoir-averaging count n on the CWE task of RULER (Qwen3-8B, 20% density). **Bold** marks the best value per column: the *highest* accuracy, and the *lowest* standard deviation over three seeds. [†] denotes our default.

n	4K	8K	16K	32K	Avg.	Std.
1	96.88	96.88	51.04	60.42	76.30	1.97
3	93.75	95.84	53.12	62.50	76.30	1.63
5 [†]	93.75	97.92	59.37	59.38	77.61	1.20
10	93.75	95.83	56.25	56.25	75.52	1.97
20	89.58	97.92	52.08	56.25	73.96	2.74

E Efficiency and Overhead Breakdown

§5.6 reports end-to-end decode throughput and steady-state memory, both of which fold the cost of Nexus Sampling into the cost of serving the retained cache. An end-to-end number cannot show whether the method is viable when the saving is smaller, because it does not separate what Nexus adds from what it removes. This appendix decomposes it at the kernel level. All measurements use Llama-3.1-8B-Instruct on a single H200 at $T_p = 8K$ prefill and 20% density, at batch size 1, and are reported per layer per decode token.

What Nexus adds. Relative to the direct-attention score that every baseline already computes, Nexus Sampling introduces two operations per eviction step. The walk of Eq. 4 runs H iterations, each one inner product $\langle \mathbf{a}^{(a)}, \mathbf{C} \rangle$ and one scaled vector add over the N_k blocks, giving $O(HN_k)$ work. The reservoir priorities of Eq. 6 draw n uniform variates per candidate block and take one power and one mean per block, giving $O(nN_k)$ work. Together these cost $4.50 \mu s$. Neither touches the value cache or the attention kernel, and both act on the block axis, where $N_k = \lceil T_k/b \rceil$ with $b = 32$. Accounting for the whole method rather than only the parts specific to it, including the windowed base score of Eq. 1, which every windowed baseline also computes, and the memory copy that compacts the cache after eviction, the total is $9.16 \mu s$.

What Nexus saves. Eviction removes two memory operations at every decode step: the attention read over the cache, and the per-step key and value append. At 20% density the resident cache is approximately 6.8 MB per layer against approximately 100 MB for dense attention. The gap is larger than the nominal density because density is set against the prefill length while the dense cache continues to grow with the decode length, as Table 5 shows: Nexus pins the working cache at a constant budget

while dense FlashAttention-2 scales as $T_p + D$.

Table 15: Kernel-level decomposition of Nexus Sampling on Llama-3.1-8B-Instruct (H200, $T_p = 8K$, 20% density, batch size 1), in μs per layer per decode token, against decode length D . *Walk + RNG* is the cost of the two stages specific to Nexus, Eqs. 4 and 6. *All overhead* additionally includes the windowed base score, which windowed baselines also compute, and the post-eviction cache compaction. Both costs are independent of decode length; the saving is not.

D	Saving	Walk + RNG		All overhead	
		Cost	Ratio	Cost	Ratio
4K	19.6	4.50	$4.4\times$	9.16	$2.1\times$
8K	24.7	4.50	$5.5\times$	9.16	$2.7\times$
16K	46.7	4.50	$10.4\times$	9.16	$5.1\times$

Net effect. Table 15 places the two against each other. The saving exceeds the walk and priority generation by $4.4\times$ to $10.4\times$, and exceeds the cost of the entire method, including the components shared with the baselines, by $2.1\times$ to $5.1\times$.

The structure of the comparison matters more than any single row. The two added stages are $O(N_k)$ vector operations whose cost is fixed by the walk depth H , the averaging count n and the block count, none of which grow with the decode length; the measured $4.50 \mu s$ is constant across the table. The saving is a reduction in the traffic of the attention step itself, and grows from 19.6 to $46.7 \mu s$ as the decode length grows from 4K to 16K. The two therefore do not scale together, and the margin widens with context length rather than narrowing.

Resource-constrained deployment. These figures are measured on an H200, so they do not directly establish behavior on a smaller accelerator. What transfers is the asymmetry rather than the absolute numbers. The overhead is a fixed per-layer per-token cost set by H , n and N_k , while the saving scales with the cache traffic the device would otherwise move, and a device with less memory bandwidth pays more for that traffic, not less. Batch size 1 is also the regime in which the saving is least favorable to us, since the fixed scoring cost is amortized over a single sequence; larger batches share the eviction decision across the batch while the traffic saving scales with it.

F Statement

We used AI assistants to assist writing. All datasets and models used in this work (LongBench (Bai

et al., 2024), RULER (Hsieh et al., 2024), Llama 3 (Team, 2024), and Qwen3 (Team, 2025)) are used solely for research evaluation, consistent with their intended use and original licenses.

G Extended Related Work

KV cache eviction vs. KV cache selection. Long-context LLM inference is bottlenecked by both the compute and the memory footprint of the KV cache, and two complementary lines of work try to address these axes. *KV cache eviction* permanently discards tokens under a fixed memory budget, addressing memory and compute simultaneously; an evicted token is irrecoverable. *KV cache selection*, often called *sparse attention* or query-aware sparsity, keeps the full cache and only chooses which tokens to attend to per query, targeting compute alone; missed tokens at step t can be re-selected at step $t + 1$. Nexus Sampling and all baselines we compare against in §5 are eviction methods; while representative sparse attention methods include **Quest** (Tang et al., 2024), **BLASST** (Yuan et al., 2026), **Sketch-and-Walk** (Le et al., 2026a), and **SOCKET** (Joshi et al., 2026). A third axis is *lossless* compression, e.g., **FAFO** (Le et al., 2026b), which drafts off a lossy compressed cache and verifies in parallel against the full cache to recover bitwise-identical generation. Nexus Sampling is orthogonal to both: any selection method can be applied on top of the surviving cache after eviction has run, and lossless schemes like FAFO still need an underlying retention policy when the full cache no longer fits.

KV cache eviction. Existing eviction methods are best understood as different choices of the *score* fed to a shared deterministic top- K selection rule, and have been comprehensively benchmarked across long-context tasks alongside other compression families by Yuan et al. (2024). **StreamingLLM** (Xiao et al., 2024) hard-codes a sink-plus-recency retention rule, motivated by the attention-sink phenomenon they identify at the first few absolute positions. **H2O** (Zhang et al., 2023) retains *heavy hitter* tokens by ranking on cumulative attention over the generation so far, formulating eviction as a dynamic submodular problem. **SnapKV** (Li et al., 2024) computes the score against an observation window of the most recent prompt tokens, which it shows are sufficient to identify the important positions for upcoming decode steps. **PyramidKV** (Cai et al., 2025) introduces

a layer-wise budget allocation that retains more tokens in lower layers and fewer in higher ones, motivated by the pyramidal information-funneling pattern of attention. **Ada-KV** (Feng et al., 2025) adapts the budget across heads instead of layers, exploiting the empirical observation that different heads concentrate their attention on different scales of the cache. **MorphKV** (Ghadia et al., 2025) maintains a constant-size cache by correlation-aware ranking against recent tokens, iteratively refining the retained set with lightweight updates. Despite the diversity of scoring strategies, every method above commits to the surviving tokens by deterministic top- K at each step, and is therefore subject to the irreversible-verdict failure mode of §3 regardless of how good its score is. Nexus Sampling is the first eviction method, to our knowledge, to replace the selection primitive itself rather than refine the score only.

Reservoir sampling. Classical reservoir sampling for streaming uniform sampling traces to **Chao** (1982) and **Vitter** (1985); **Efrimidis and Spirakis** (2006) extended the scheme to weighted-without-replacement sampling via the priority u_j^{1/w_j} , which is exactly the per-step primitive Nexus uses. We instantiate this rule as an eviction primitive and add the n -fold averaged variant of §2.3, which interpolates between unbiased ($n = 1$) and effectively deterministic ($n \rightarrow \infty$) retention with a single knob, recovering the existing deterministic-top- K template as a limit case.

Connection to Sketch-and-Walk. Le et al. (2026a) introduce a multi-hop walk for sparse *attention* over a full cache, propagating block importance *across layers* by composing per-layer block-attention matrices through an iterative recurrence. The Nexus walk of §2.2 is structurally analogous but operates on an orthogonal dimension: it propagates importance *across tokens*, using the per-window per-query distribution to expose bridge tokens that no single recent query attends to strongly. The two are therefore complementary rather than competing, addressing two stages of the same pipeline: Sketch-and-Walk reduces the cost of reading from a still-full cache, while Nexus Sampling decides which tokens survive into that cache.