SANTA++: SAMPLING ATTENTION THROUGH REPRESENTATIVE KEYS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention often concentrates on a small subset of tokens in the context, but which subset matters changes from one query to the next. To exploit this changing structure, we introduce SANTA++, a training-free stochastic attention method that uses representative keys for memory-efficient selection without scanning the entire key-value (KV) cache. Cached keys are organized into teams, and the query scores one representative from each team to decide which teams to sample. We compute exact attention scores within the sampled teams and reweight each team's contribution by the inverse of its inclusion probability. This importance sampling correction estimates attention over the full cache, with a sampling budget that lets us trade memory reads for accuracy. Remarkably, with 32 or 64 sampled teams, SANTA++ uses 16% to 22% of dense attention's KV reads and retains 94% to 99% of the dense-attention baseline's scores on LongBench v2 and HELMET's retrieval-augmented generation subset, and 85% to 91% on RULER, with Qwen2.5-7B-Instruct at 32K context. With 31 sampled teams, our GPU implementation delivers a $1.69\times$ attention speedup over the dense FlashAttention baseline at 32K context. By reducing the number of cache entries read, SANTA++ in principle complements architectures with compressed KV representations, such as multi-head latent attention. Our kernels are available at: https://anonymous.4open.science/r/santapp-triton-demo-020D.

1 INTRODUCTION

Which cached tokens make the largest contributions to attention depends on the current decoding query. At long context lengths, attention can concentrate on a small subset of cached tokens (Zhang et al., 2023; Tang et al., 2024), but identifying that subset from exact attention scores still requires reading every key. Selecting fewer tokens then saves value reads while leaving a full-key scan. When decoding is limited by memory bandwidth, these key-value (KV) cache reads are costly, and their cost grows with context length (Ribar et al., 2024). For example, with Qwen2.5-7B-Instruct at a 32K-token context, streaming bf16 keys and values amounts to about 64 MiB per layer for each generated token. To reduce both selection and evaluation reads, a sparse method must find useful entries without first scoring every key.

We introduce SANTA++, a training-free attention method that routes queries through actual-key representatives (Figure 1). After dense prefill, we divide prompt keys into parent groups, choose representative keys within each parent, and assign the remaining keys to their nearest representative. Each resulting team contains its representative. For each decoding query, we use each representative score and team size to estimate the team's attention mass, then sample teams according to these estimates. We then read the selected teams' keys and values and evaluate every selected member's exact query–key score. Newly generated tokens form a separate suffix evaluated exactly. SANTA constructs the exact attention distribution by scoring every cached key, then samples from that distribution (Lee et al., 2026). In the limiting case where every key forms its own team, each representative is the key itself, and SANTA++ recovers this exact-attention sampling distribution. SANTA++ can therefore be viewed as a grouped generalization of SANTA, replacing the full-key scan with approximate routing to enable substantially lower KV access.

Actual-key representatives avoid relying on a group's exponentiated mean-key score to estimate its attention mass. Averaging logits before exponentiation can suppress a high-scoring member's con-

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

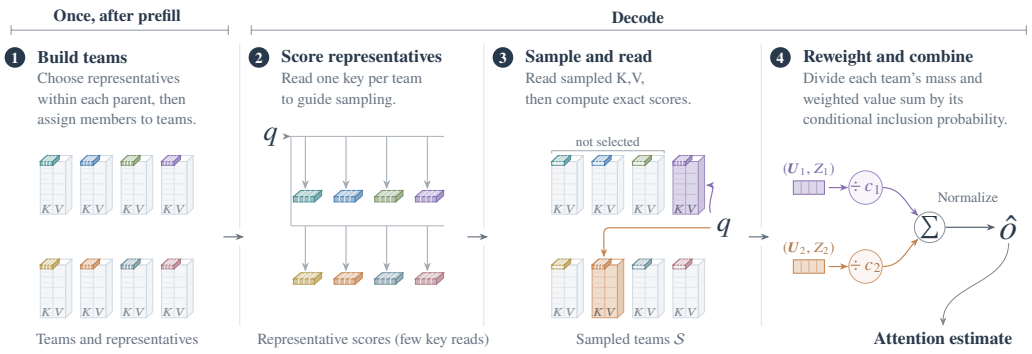


Figure 1: **Representative-guided attention sampling.** After prefill, parent groups cover the prompt, and actual member keys represent the teams built within each parent. For each decoding query, we form a routing weight for each team from its representative score and team size, and use these weights to sample teams $\mathcal{S}$. Selected teams' keys and values are read in full, and their members' exact attention contributions are computed. Each team's unnormalized attention mass and weighted value sum are divided by its conditional inclusion probability c_g before the corrected sums are combined and normalized (Section 2).

tribution. Representative scores guide selection, while exact member scores determine the selected teams' contributions. Because teams have unequal chances of selection, we divide each selected team's unnormalized attention mass and weighted value sum by its conditional inclusion probability before normalization (Section 2). This correction targets full-cache contributions rather than simply renormalizing an uncorrected selected subset.

The nominal budget S controls how many teams are selected, providing an adjustable accuracy–KV-access trade-off. Actual access also depends on team sizes, routing and suffix reads, and overlap across query heads. We form parent groups either with k-means or from contiguous token spans. K-means groups keys by similarity. Contiguous spans instead define parents directly from token positions, avoiding the k-means preparation step. Both use the same distance-based team construction, and every prompt team remains eligible for each query. For the Qwen2.5-7B-Instruct evaluations below, we fix the nominal parent size to $P = 16$ and use up to $R = 4$ representatives per parent. These values were chosen heuristically and kept fixed across benchmarks, budgets, and the k-means and contiguous grouping policies.

Our evaluation with Qwen2.5-7B-Instruct at 32K context shows that SANTA++ can retain most of dense attention's accuracy while reading much less of the KV cache. On LongBench v2 (Bai et al., 2025), it retains 99.07% of the dense baseline's score while using 16.68% of its KV reads. On HELMET's retrieval-augmented generation subset (Yen et al., 2025), it retains 98.00% of the dense score at 38.49% KV access. On RULER (Hsieh et al., 2024), increasing the sampling budget raises the retained score from 91.07% to 98.54% while KV access increases from 19.95% to 46.95%. Contiguous grouping avoids the k-means preparation step and remains competitive in several settings, providing a simpler alternative when clustering cost matters.

Separately, our Triton implementation gives a $1.69\times$ attention-operator speedup over Flash SDPA on a 32K-context, batch-one, FP16 workload from layer 14 of Qwen2.5-7B-Instruct on an NVIDIA GeForce RTX 5090 Laptop GPU (Section 4). The comparison uses 31 sampled teams per query head (nominal budget $S = 124$), an implementation-specific setting described in Section 4. Appendix F tests the method's applicability to Multi-Head Latent Attention (MLA), and whether useful token sparsity remains in DeepSeek-V2-Lite-Chat after latent compression and head sharing.

2 Method

At long context lengths, attention can be concentrated on a small subset of cached tokens, but selecting that subset by its exact attention logits requires reading every key. To reduce these selection

reads, SANTA++ makes its routing decisions from one representative key per team rather than scoring every prompt key.

For one decoding query $\boldsymbol{q} \in \mathbb{R}^d$, let $\mathcal{C}$ index the currently cached key-value pairs $(\boldsymbol{k}_i, \boldsymbol{v}_i)$. The keys and query include the model's positional embedding. The full-cache attention output is

$$s_i = \frac{\boldsymbol{q}^\top \boldsymbol{k}_i}{\sqrt{d}}, \quad Z = \sum_{i \in \mathcal{C}} e^{s_i}, \quad \boldsymbol{U} = \sum_{i \in \mathcal{C}} e^{s_i} \boldsymbol{v}_i, \quad \boldsymbol{o} = \frac{\boldsymbol{U}}{Z}. \tag{1}$$

Here Z is the full-cache unnormalized attention mass, and $\boldsymbol{U}$ is its value-weighted numerator. SANTA++ approximates these two sums by sampling teams and evaluating all members of the selected teams (Figure 1). Representatives guide which teams to read, and they do not replace the selected members' keys or values in the attention calculation.

2.1 Approximating Group Mass

A single mean key is a convenient summary for routing, but it gives a group's mean logit rather than its attention mass. For a group $\mathcal{G}$ of n keys, write $\bar{\boldsymbol{k}} = n^{-1} \sum_{i \in \mathcal{G}} \boldsymbol{k}_i$ and $\bar{s} = \boldsymbol{q}^\top \bar{\boldsymbol{k}} / \sqrt{d}$. Convexity of the exponential gives

$$\underbrace{n e^{\bar{s}}}_{\text{centroid mass surrogate}} \leq \underbrace{\sum_{i \in \mathcal{G}} e^{s_i}}_{\text{exact group mass}}. \tag{2}$$

Because the surrogate exponentiates the group's mean logit rather than exponentiating each member before summing, a large member logit is averaged with the others, while the exact sum retains its full contribution. Figure 2 illustrates the consequence: 16 constructed logits give an exact mass of approximately 2.20×10^4, compared with approximately 29 for the surrogate.

Equality holds when all within-group logits are equal. The bound applies to unnormalized mass, not to each normalized group probability. Different gaps across groups can change their relative routing weights and hence which groups are likely to be selected. Appendix B.1 gives additional examples.

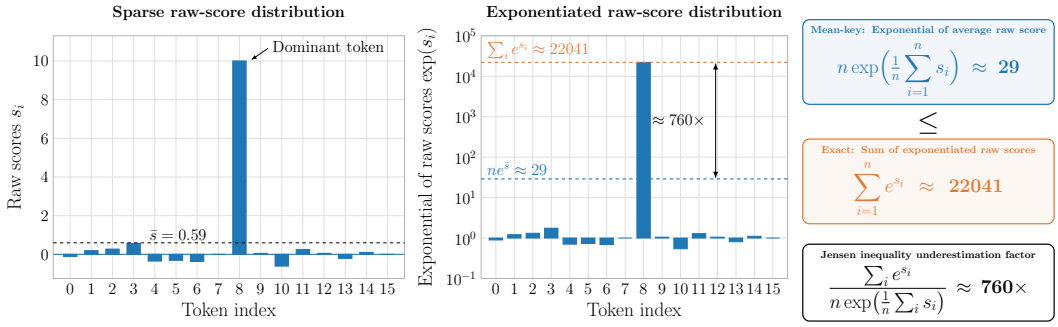


Figure 2: **A mean-key mass surrogate for 16 constructed logits.** One high score dominates the exact mass, which is approximately $760\times$ the surrogate.

2.2 Preparing Parent Groups and Teams

SANTA++ divides each parent group into teams represented by actual keys. This lets a query choose among different parts of the parent instead of assigning all its members one mean-key routing weight.

Parent groups restrict the distance comparisons needed to build teams. After dense prefill, we form them separately for each layer and KV head from the N prompt keys after rotary position encoding (RoPE) (Su et al., 2024), using either k-means on the keys or consecutive spans of P tokens. Here P is a nominal parent size. K-means parent sizes can vary. Contiguous parents avoid clustering while still covering the full prompt. Both alternatives use the same team construction, in which each member is compared only with representatives from its own parent. Our distance diagnostic finds

that contiguous groups are more compact than random groups, although less compact than k-means groups (Appendix B.3).

Within each parent, we choose up to R actual keys as representatives. The first is nearest the parent's mean key. Each later choice is farthest from its nearest existing representative, to cover members poorly represented so far. The mean thus helps choose the first actual key rather than serving as the routing key itself. Every member joins its nearest representative's team, and each representative belongs to its own team. Distances are squared Euclidean distances on the unnormalized keys. Contiguous parents need not yield contiguous teams, because assignment uses key distance rather than token order.

The resulting M nonempty teams $\mathcal{T}_g$ partition all prompt rows. We store each team's representative $\boldsymbol{\ell}_g$, size n_g, and member indices. Parent formation and team construction are performed once after prefill, and the resulting records are reused throughout decoding. Their preparation cost is therefore spread over the generated tokens. Every prompt team remains eligible at each query. Newly generated rows form a separate suffix $\mathcal{E}$ evaluated exactly. Appendices B.4 and B.5 give additional diagnostics on contiguous windows and key-versus-probe clustering, respectively.

2.3 Sampling and correcting team contributions

The routing weight should reflect both query relevance and team size. We compute it using one representative key per team:

$$\widetilde{Z}_g = n_g \exp\left(\frac{\boldsymbol{q}^\top \boldsymbol{\ell}_g}{\sqrt{d}}\right), \qquad \phi_g = \log \widetilde{Z}_g. \tag{3}$$

The factor n_g accounts for how many members the representative describes. We add independent Gumbel noise to ϕ_g and select the K teams with the largest perturbed routing scores using the Gumbel-top-K trick (Kool et al., 2019), forming a set $\mathcal{S}$ sampled globally without replacement. The nominal budget S determines K but is not a row count. With $P = 16, R = 4, S = 128$ requests 32 teams. The selected prompt-row count is $\sum_{g\in\mathcal{S}} n_g$, which can vary. Representative routing reads and exact-suffix reads are additional to those selected rows. Appendix A gives the budget rules and sampler.

After selection, we read the keys and values of every member of each selected team and evaluate its exact contribution: $Z_g = \sum_{i\in\mathcal{T}_g} e^{s_i}$ and $\boldsymbol{U}_g = \sum_{i\in\mathcal{T}_g} e^{s_i}\boldsymbol{v}_i$. Thus an approximate representative score decides what to read.

Without correction, frequently selected teams would receive too much weight relative to teams selected less often. We therefore divide both a selected team's mass and its weighted value sum by the same inclusion correction before normalization. For a selected team, c_g is its inclusion probability conditional on the other teams' perturbed routing scores and the current query, cache, team records, and budget. Appendix A derives the conditional correction and gives its expression in Equation 7. Defining $Z_\mathcal{E}$ and $\boldsymbol{U}_\mathcal{E}$ by the same sums over the exact suffix, we use

$$\widehat{\boldsymbol{U}} = \boldsymbol{U}_\mathcal{E} + \sum_{g\in\mathcal{S}} \frac{\boldsymbol{U}_g}{c_g}, \quad \widehat{Z} = Z_\mathcal{E} + \sum_{g\in\mathcal{S}} \frac{Z_g}{c_g}, \quad \widehat{\boldsymbol{o}} = \frac{\widehat{\boldsymbol{U}}}{\widehat{Z}}. \tag{4}$$

All members receive the same correction because the team is the sampling unit. The two unnormalized sums are unbiased (Equation 9 in Appendix A). Their ratio, which gives the attention output, is generally biased. Selecting all teams, with $c_g = 1$, recovers dense attention.

For grouped-query attention (GQA), query heads associated with the same KV head use the same team partition but make independent selections and retain their own inclusion corrections. Keys and values are read for the union of those selections, while each team contributes only to the heads that selected it.

2.4 Attention-distribution diagnostic

Inclusion correction accounts for unequal selection chances, but routing still determines which teams are likely to be read at a limited budget. We therefore use Kullback–Leibler (KL) divergence to

compare how closely each routing distribution follows dense attention over the same prompt tokens. We spread each group's routing probability equally across its members and compare the resulting token distribution with dense attention over the prompt.

As a diagnostic baseline, we compare SANTA++'s actual-key representatives with centroid-guided sampling, where each k-means group is routed using its arithmetic-mean key. We evaluate two nominal group sizes for this baseline, $B = 4$ and $B = 16$. The $P = 16, R = 4$ team variant with k-means parents has lower mean KL divergence than both centroid-guided variants on all three tasks (Table 1). All variants use the same queries and cache states from dense SDPA generation with Qwen2.5-7B-Instruct at 8,192-token context, with 20 prompts per task. This diagnostic measures routing agreement with dense attention before sampling. Appendix B.2 defines the distributions and averaging procedure and reports the separate 100-prompt-per-task accuracy comparison.

Task	Teams (k-means) $P = 16, R = 4$	Centroid-guided sampling $B = 4$	Centroid-guided sampling $B = 16$
CWE	1.621	2.806	3.429
S-NIAH-1	1.248	2.076	2.827
QA-1	1.613	2.429	3.495

Table 1: **Routing approximation to dense attention.** Mean $\mathrm{KL}(p\|q)$ in nats, where p is dense attention over prompt tokens and q spreads each group's routing probability equally across its members. All variants use the same dense SDPA queries and cache states. No sampling is performed in this diagnostic. Appendix B.2 gives the definitions and averaging procedure.

3 Accuracy and theoretical KV access

SANTA++ reduces reported logical KV access below SANTA's full-key-scan cost at useful operating points across three benchmarks. We compare dense scaled dot-product attention (SDPA), SANTA (Lee et al., 2026), which samples from exact attention weights after a full-key scan, and SANTA++ with k-means or contiguous grouping using Qwen2.5-7B-Instruct (Qwen Team, 2024). All main results use a 32K context budget and cover HELMET retrieval-augmented generation (RAG) (Yen et al., 2025), LongBench v2 (Bai et al., 2025), and RULER (Hsieh et al., 2024). Their complete 32K aggregate sweeps are reported in Tables 2, 3, and 4, respectively. Evaluation setup and access accounting are given in Appendix C.1, with additional HELMET, LongBench v2, and RULER results in Appendices C.2, C.3, and C.4, respectively.

KV denotes reported logical key and value vector access relative to dense decoding. For full clarity, these tables do not report measured DRAM traffic, resident cache size, or latency. Under the reference counter's equal costs for key and value vectors, SANTA's full-key scan contributes 50% before sampled value reads. For SANTA++, team sizes and selected-row overlap across query heads sharing a KV head affect access, so the nominal budget S is not a unique-row count. Equal S therefore does not impose equal access. Appendix C.1 gives the counter and aggregation details.

HELMET RAG. At 32K context, SANTA++ with k-means retains 98.00% of the SDPA RAG score at 38.49% KV access ($S = 1024$; Table 2). A smaller budget of $S = 128$ retains 94.43% at 17.49% access, providing a lower-access option.

Contiguous grouping omits parent k-means while retaining the same team construction within each parent. At $S = 1024$, it retains 97.56% of the SDPA score at 31.23% access. Its absolute score is 56.00%, compared with 56.25% for k-means: a difference of 0.25 percentage points with 7.26 percentage points less access, while also avoiding the parent clustering step.

The RAG score and access summaries weight the four RAG tasks equally. Access includes representative-key routing reads as well as selected KV reads. Appendix C.2 reports the other HELMET tasks separately because their metrics differ. Not all of them are as robust as RAG: at 32K, JSON KV retrieval falls from 96.80% under SDPA to 45.60% with k-means at S = 128 (33.40% with contiguous grouping), recovering to 87.00% at $S = 512$ (Table 10).

Table 2: **HELMET RAG: full 32K budget sweep with Qwen2.5-7B-Instruct.** Score (abs) is the equal-weight mean substring exact match on Natural Questions, TriviaQA, PopQA, and HotpotQA. Score (norm) is the absolute score divided by the corresponding SDPA score at the same context length, expressed as a percentage. KV is the equal-weight mean of the four RAG task-level total-access percentages, including representative-key routing reads. S is the nominal budget. The $\pm$ entries are 95% bootstrap confidence interval half-widths.

Method	S	KV (%)	Score (norm) (%)	Score (abs) (%)
SDPA	–	100.00	100.00	57.40 ± 2.00
SANTA	128	50.43	99.13	56.90 ± 2.05
	256	50.75	99.04	56.85 ± 2.05
	512	51.30	100.17	57.50 ± 2.02
	1024	52.17	99.48	57.10 ± 2.07
	2048	53.53	99.65	57.20 ± 2.08
SANTA++ (k-means)	128	17.49	94.43	54.20 ± 2.05
	256	21.82	95.56	54.85 ± 2.03
	512	28.58	96.43	55.35 ± 2.05
	1024	38.49	98.00	56.25 ± 2.02
	2048	52.13	99.22	56.95 ± 2.07
SANTA++ (contiguous)	128	15.66	92.51	53.10 ± 2.05
	256	18.49	94.77	54.40 ± 2.05
	512	23.35	95.03	54.55 ± 2.07
	1024	31.23	97.56	56.00 ± 2.07
	2048	43.34	97.30	55.85 ± 2.05

LongBench v2. With a 32K context cap, SANTA++ with k-means retains 99.07% of the SDPA score at 16.68% reported KV access ($S = 128$; Table 3). Its absolute accuracy is 31.61%, compared with 31.91% for SDPA, a difference of 0.30 percentage points. This configuration uses 66.9% less reported access than SANTA's lowest-access tested setting, which uses 50.36%. Near-dense accuracy is therefore possible well below the access cost of a full-key scan. The k-means and SANTA sweeps do not show a consistent increase in accuracy as access grows.

To avoid the preparation cost of parent k-means, contiguous grouping forms parents directly from consecutive token spans, then applies the same distance-based team construction within each parent. At 15.27% reported access ($S = 128$), it reaches 30.91% absolute accuracy, 0.70 percentage points below the k-means configuration above while using slightly less access. Increasing access to 22.15% ($S = 512$) raises its absolute accuracy to 31.41%, or 98.44% of the SDPA score. Thus, on LongBench v2, retaining a near-dense score at less than one-quarter of dense KV access does not require parent k-means.

RULER. At 32K, SANTA++ with k-means retains 91.07% of the SDPA aggregate score at 19.95% reported KV access ($S = 256$; Table 4). Increasing access to 25.70% ($S = 512$) raises the retained score to 93.99%, providing a higher-accuracy option while still using roughly one-quarter of dense access. For comparison, SANTA's lowest-access tested setting retains 99.15% of the SDPA score at 50.32% access. The 19.95%-access SANTA++ configuration reduces reported access by 60.4% relative to that setting.

Contiguous grouping removes the parent k-means preparation step and offers a similar aggregate score at a nearby access level. At 21.37% access ($S = 512$), it reaches an absolute aggregate score of 75.37%, compared with 76.05% for k-means at 19.95% access. For these configurations, omitting parent k-means gives a score 0.68 percentage points lower with 1.42 percentage points more access. At 28.16% access ($S = 1024$), contiguous grouping reaches 79.40% absolute score, retaining 95.07% of the SDPA aggregate. The simpler parent formation therefore remains useful below 30% access, without requiring parent clustering.

The aggregate includes all 13 RULER tasks, with 500 completed prompts per task. Table 12 shows that accuracy retention varies across tasks. For contiguous grouping at $S = 1024$, the three S-NIAH tasks each score at least 98.80%, but MK-NIAH-3 scores 60.20%, compared with 89.60% for SDPA.

Table 3: **LongBench v2: full budget sweep with a 32K context cap.** Qwen2.5-7B-Instruct is evaluated on 503 prompts using the brief-reasoning prompt and middle-truncation procedure in Appendix C.3. Score (abs) is accuracy. Score (norm) is the absolute score divided by the corresponding SDPA score, expressed as a percentage. KV is reported GQA-aware logical access, and S is the nominal budget. The $\pm$ entries are the reported 95% bootstrap confidence interval half-widths.

Method	S	KV (%)	Score (norm) (%)	Score (abs) (%)
SDPA	—	100.00	100.00	31.91 $\pm$ 2.04
SANTA	128	50.36	95.95	30.62 $\pm$ 2.06
	256	50.59	102.80	32.80 $\pm$ 1.96
	512	50.97	106.85	34.10 $\pm$ 2.04
	1024	51.56	99.84	31.86 $\pm$ 1.91
SANTA++ (k-means)	128	16.68	99.07	31.61 $\pm$ 2.01
	256	20.41	95.33	30.42 $\pm$ 1.96
	512	26.46	101.71	32.46 $\pm$ 2.09
	1024	35.62	99.38	31.71 $\pm$ 2.04
SANTA++ (contiguous)	128	15.27	96.88	30.91 $\pm$ 2.09
	256	17.74	97.35	31.06 $\pm$ 2.01
	512	22.15	98.44	31.41 $\pm$ 1.96
	1024	29.52	98.60	31.46 $\pm$ 2.04

Table 4: **RULER: full 32K budget sweep with Qwen2.5-7B-Instruct.** Score (abs) is the equal-weight mean of all 13 task scores over 500 completed prompts per task. Score (norm) is the absolute score divided by the corresponding SDPA score at the same context length, expressed as a percentage. KV is pooled GQA-aware logical access, including representative-key routing reads, and S is the nominal budget. Absolute-score $\pm$ entries are 95% task-stratified bootstrap confidence interval half-widths.

Method	S	KV (%)	Score (norm) (%)	Score (abs) (%)
SDPA	—	100.00	100.00	83.51 $\pm$ 0.66
SANTA	128	50.32	99.15	82.80 $\pm$ 0.64
	256	50.54	99.75	83.31 $\pm$ 0.67
	512	50.91	100.11	83.61 $\pm$ 0.65
	1024	51.52	99.92	83.45 $\pm$ 0.67
	2048	52.49	99.99	83.50 $\pm$ 0.66
	4096	53.96	99.91	83.44 $\pm$ 0.67
SANTA++ (k-means)	128	16.38	85.41	71.33 $\pm$ 0.74
	256	19.95	91.07	76.05 $\pm$ 0.74
	512	25.70	93.99	78.49 $\pm$ 0.74
	1024	34.44	96.27	80.40 $\pm$ 0.72
	2048	46.95	98.54	82.29 $\pm$ 0.66
	4096	63.65	99.48	83.08 $\pm$ 0.64
SANTA++ (contiguous)	128	15.03	77.50	64.72 $\pm$ 0.77
	256	17.31	84.46	70.54 $\pm$ 0.76
	512	21.37	90.25	75.37 $\pm$ 0.75
	1024	28.16	95.07	79.40 $\pm$ 0.73
	2048	38.89	97.49	81.42 $\pm$ 0.72
	4096	54.68	98.86	82.56 $\pm$ 0.67

4 GPU IMPLEMENTATION AND ATTENTION LATENCY

SANTA++ reduces complete attention-operator latency from $107.33\,\mu$s for Flash SDPA to $63.52\,\mu$s, a $1.69\times$ speedup on a single-token decode workload from layer 14 of Qwen2.5-7B-Instruct (Table 5). The workload has 32,768 cached tokens, batch size one, and FP16 inputs, and runs on an NVIDIA GeForce RTX 5090 Laptop GPU. SANTA++ selects $K = 31$ teams per query head (nominal budget $S = 124$), with no appended suffix. This value is implementation-driven: the sampler retains the top K teams plus the next-highest Gumbel-perturbed routing score needed to compute the conditional-inclusion threshold. Thus $K = 31$ gives a power-of-two working set of 32 scores for the Triton top-K stage.

The implementation scores actual-key representatives rather than all cached keys during selection. Under grouped-query attention (GQA) (Ainslie et al., 2023), it reuses both representative-key loads and selected-member key and value loads across query heads that share a KV head. Each head keeps its own selected teams and conditional inclusion corrections, and accumulates contributions only from teams it selected. Representatives guide selection, while the selected members' exact scores determine their contributions. Inverse-inclusion weights apply to both the unnormalized attention mass and the weighted value sum. Appendix D describes the five kernel programs.

The dense baseline is PyTorch SDPA using FlashAttention (Dao, 2024) with native GQA enabled. We time the complete attention operation, including routing, selection, and all five kernel launches. Preparation, including cache packing and allocation of temporary buffers, is completed before timing. End-to-end gains also depend on preparation cost and generation length.

The kernel benchmark uses $S = 124$, while the task-accuracy sweeps start at $S = 128$. Appendix E gives the complete kernel profiling protocol.

Table 5: Complete attention-operator latency on the captured 32K, batch-one FP16 workload. Measurements include selection and all launches. Preparation is excluded.

Method	GPU launches	Median latency (μs)	Speedup vs. Flash
SANTA++	5	**63.52**	**1.69**$\times$
Forced Flash SDPA	2	107.33	1.00$\times$

5 RELATED WORK

Compressed representations. Quantization (Liu et al., 2024; Frantar et al., 2023) and KV sharing or latent compression (Ainslie et al., 2023; DeepSeek-AI et al., 2024) reduce the memory occupied by model weights or the KV–cache. SANTA++ targets a complementary cost: how many cached entries each decoding query reads. Reducing these reads can save memory traffic even when the cache already uses compact representations.

Sparse attention and cache retention. Sparse attention methods reduce computation by evaluating or approximating only part of the full attention computation, using structured patterns (Child et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020), low-rank projections (Wang et al., 2020), hashing (Kitaev et al., 2020), or deterministic top-k selection (Gupta et al., 2021). SANTA++ instead treats sparse decode-time access as an estimation problem: it uses query-dependent sampling and corrects the sampled contributions toward the full-cache attention sums. Cache-retention methods such as H_2O, StreamingLLM, and DuoAttention (Zhang et al., 2023; Xiao et al., 2024b; 2025) reduce memory use by limiting which KV entries remain available. SANTA++ does not require permanently dropping prompt tokens, so entries skipped for one query can still be selected later. For very long contexts where some eviction is acceptable, the two approaches can also be combined: cache retention reduces storage, while SANTA++ reduces the memory traffic from reading the retained cache.

Query-aware retrieval and grouping. Query-aware methods use approximate scores or group summaries to identify promising parts of the cache (Ribar et al., 2024; Tang et al., 2024; Sun et al.,

2025; Hao et al., 2025). InfLLM (Xiao et al., 2024a), for example, uses selected tokens to represent contiguous memory blocks, while centroid-based methods use group means for retrieval, merging, or approximation (Liu et al., 2025; Hooper et al., 2025b; Hu et al., 2026; Hooper et al., 2025a). SANTA++ builds on the same general idea of routing through a small group summary, but our analysis identifies a limitation of using the arithmetic-mean key for this purpose. By Jensen's inequality, scoring a mean key systematically underestimates the group's unnormalized attention mass (Equation 2). This motivates our use of actual keys as representatives rather than mean keys. This observation may also inform centroid-based retrieval methods, where replacing or correcting mean-key scores could better preserve group attention mass.

Sampling and importance correction. Sampling-based attention approximations have also been explored (Kim & Ko, 2022). SANTA (Lee et al., 2026) samples from the exact post-softmax attention distribution, which requires scoring every cached key before sampling. SANTA++ can be viewed as a grouped generalization of this idea: representative scores and team sizes approximate each team's attention mass, replacing the full-key scan with group-level routing. In the limit where the number of teams equals the context length, every key forms its own team, and SANTA++ recovers SANTA's exact sampling distribution. With larger teams, SANTA++ trades exact routing for lower KV access. MagicPIG (Chen et al., 2025) is closely related in spirit: it also treats sparse attention as an estimation problem rather than simply renormalizing over a selected subset, using locality-sensitive hashing for self-normalized importance sampling. SANTA++ develops the same estimator view through explicit groups, using representative estimates to allocate samples across the cache and correcting the selected contributions for their sampling probabilities.

6 Conclusion

SANTA++ exploits query-dependent attention sparsity without scanning the full KV cache. It uses actual-key representatives to route queries to teams, computes exact attention scores within the selected teams, and corrects their contributions for unequal inclusion probabilities. This provides an adjustable trade-off between attention accuracy and KV access without retraining.

Under a 32K context budget, SANTA++ with k-means retains 99.07% of the SDPA score on LongBench v2 at 16.68% logical KV access and 98.00% on HELMET RAG at 38.49% access. On RULER, a larger budget retains 98.54% at 46.95% KV access. Contiguous grouping provides a simpler alternative when the preparation cost of k-means is undesirable: on HELMET RAG, it retains 97.56% of the SDPA score at 31.23% access while avoiding k-means clustering entirely. Our Triton implementation reduces attention-operator latency from $107.33\,\mu$s to $63.52\,\mu$s on the evaluated 32K workload, a $1.69\times$ speedup.

The DeepSeek-V2-Lite-Chat experiments in Appendix F further suggest that useful token sparsity can remain after latent compression and head sharing. This motivates combining SANTA++'s selective token access with MLA-style architectures to reduce memory traffic in already compressed models.

AI Use Statement

We used generative AI tools to assist with writing and presentation, software implementation and debugging, methodological and mathematical development, and the analysis and interpretation of experimental results. All AI-assisted outputs were reviewed and validated by the authors.

Reproducibility statement

We release our GPU kernels as open-source software on GitHub, linked in the abstract.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*

Processing, pp. 4895–4901. Association for Computational Linguistics, December 2023. URL https://aclanthology.org/2023.emnlp-main.298/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664. Association for Computational Linguistics, July 2025. URL https://aclanthology.org/2025.acl-long.183/.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations*, volume 2025, pp. 44169–44190, 2025. URL https://openreview.net/forum?id=ALzTQUgW8a.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

DeepSeek-AI et al. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In Nafise Sadat Moosavi, Iryna Gurevych, Angela Fan, Thomas Wolf, Yufang Hou, Ana Marasović, and Sujith Ravi (eds.), *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pp. 39–52, Virtual, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.sustainlp-1.5. URL https://aclanthology.org/2021.sustainlp-1.5/.

Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. OmniKV: Dynamic context selection for efficient long-context LLMs. In *International Conference on Learning Representations*, volume 2025, pp. 87443–87464, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/da1131a86ac3c70e0b7cae89c3d4df22-Paper-Conference.pdf.

Coleman Hooper, Sebastian Zhao, Luca Manolache, Sehoon Kim, Michael Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Multipole attention for efficient long context reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 37382–37405. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1255. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/359f442022ef31bcd9e41959576107ef-Paper-Conference.pdf.

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

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

Jie Hu, Shengnan Wang, Yutong He, Ping Gong, Jiawei Yi, Juncheng Zhang, Youhui Bai, Renhai Chen, Gong Zhang, Cheng Li, and Kun Yuan. CentroidKV: Efficient long-context LLM inference via KV cache clustering. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=T3EeupQhGj.

Hyunjun Kim and JeongGil Ko. Fast monte-carlo approximation of the attention mechanism. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 7185–7193, 2022.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pp. 3499–3508. PMLR, 2019.

Kyle Lee, Corentin Delacour, Kevin Callahan-Coray, Kyle Jiang, Can Yaras, Samet Oymak, Tathagata Srimani, and Kerem Yunus Camsari. Stochastic sparse attention for memory-bound inference. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ptaIjzIuoY.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=L057s2Rq8O.

Qwen Team. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient LLM inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42558–42583. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ribar24a.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57355–57373. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/sun25b.html.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=bTHFrqhASY.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cFu7ze7xUm.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, volume 2025, pp. 98914–98965, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f5332c8273d02729730a9c24dec2135e-Paper-Conference.pdf.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

A Algorithm and estimator details

We separate parent formation, team construction, and query-dependent sampling. Algorithms 3 and 4 provide alternative prompt partitions. Both use the same team builder (Algorithm 2) and decode estimator (Algorithm 1).

Cache state and budget. For one layer and KV head, let $\boldsymbol{k}_i^{\mathrm{pre}}$ be the key saved during dense prefill. Queries and keys are post-RoPE; construction uses raw keys without normalization or feature standardization. The M nonempty teams $\mathcal{T}_g$ partition the N prompt indices, $n_g = |\mathcal{T}_g|$, and $\boldsymbol{\ell}_g$ is an actual prefill key from team g. Memberships and representative keys remain fixed after prefill. The current cache is the disjoint union of the prompt teams and the exact generated suffix $\mathcal{E}$, with only query-visible positions. Query heads sharing a KV head share team records, but sample independently with their own thresholds and corrections. Reading their union does not add another head's selections to a head's estimator.

For positive integer parameters P, R, S, require P to be divisible by R and S to be divisible by P/R. The selected-team count is

$$K = \min\left\{M, \frac{S}{P/R}\right\}. \tag{5}$$

The number of selected prompt rows is $\sum_{g\in\mathcal{S}} n_g$; this excludes representative-routing reads and the exact suffix. After t generated tokens have been fed back, the suffix is $\mathcal{E} = \{N+1, \ldots, N+t\}$. It

is empty for the first sparse prediction, which the reference obtains by recomputing the final prompt token rather than using dense-prefill logits. No prompt window is reserved for exact evaluation, including a short final contiguous parent.

Algorithm 1 SANTA++ attention for one query head

Require: Query $\boldsymbol{q}$; current KV cache; team records $(\mathcal{T}_g, \boldsymbol{\ell}_g)_{g=1}^{M}$; exact indices $\mathcal{E}$; requested integer team count $K \geq 1$.
Ensure: Estimated attention output $\widehat{\boldsymbol{o}}$.

$K \leftarrow \min(K, M)$
if $K = M$ **then**
  $\mathcal{S} \leftarrow \{1, \ldots, M\}$ and $c_g \leftarrow 1$ for every $g \in \mathcal{S}$
else
  $\phi_g \leftarrow \log n_g + \boldsymbol{q}^\top \boldsymbol{\ell}_g / \sqrt{d}$ for $g = 1, \ldots, M$
  Draw independent $G_g \sim \mathrm{Gumbel}(0, 1)$ and set $Y_g \leftarrow \phi_g + G_g$ for every g
  $\mathcal{S} \leftarrow$ indices of the K largest perturbed routing scores among $Y_1, \ldots, Y_M$
  $\tau \leftarrow$ the $(K+1)$st-largest perturbed routing score among $Y_1, \ldots, Y_M$
  $c_g \leftarrow 1 - \exp\!\big[-\exp(\phi_g - \tau)\big]$ for every $g \in \mathcal{S}$
end if
$\mathcal{I} \leftarrow \mathcal{E} \cup \bigcup_{g \in \mathcal{S}} \mathcal{T}_g$
for each $i \in \mathcal{I}$ **do**
  Read $(\boldsymbol{k}_i, \boldsymbol{v}_i)$ and compute $s_i \leftarrow \boldsymbol{q}^\top \boldsymbol{k}_i / \sqrt{d}$, $a_i \leftarrow e^{s_i}$
end for
$\widehat{\boldsymbol{U}} \leftarrow \sum_{i \in \mathcal{E}} a_i \boldsymbol{v}_i + \sum_{g \in \mathcal{S}} \frac{1}{c_g} \sum_{i \in \mathcal{T}_g} a_i \boldsymbol{v}_i$
$\widehat{Z} \leftarrow \sum_{i \in \mathcal{E}} a_i + \sum_{g \in \mathcal{S}} \frac{1}{c_g} \sum_{i \in \mathcal{T}_g} a_i$
return $\widehat{\boldsymbol{o}} = \widehat{\boldsymbol{U}} / \widehat{Z}$

Conditional inclusion correction. We use the threshold-based inclusion correction for Gumbel-top-K sampling (Kool et al., 2019). For $1 \leq K < M$, condition on the state $\mathcal{D}$ before sampling: the current query, cache, team records, exact suffix, and budget. Assume finite logits and values, positive routing weights, and fresh independent standard Gumbel draws conditional on $\mathcal{D}$, with cumulative distribution $\Pr(G_g \leq x) = \exp[-\exp(-x)]$. The perturbed routing scores $Y_g = \phi_g + G_g$ then have no ties with probability one. The ordered sampling law chooses among remaining teams in proportion to $\widetilde{Z}_g$, so $p_g = \widetilde{Z}_g / \sum_h \widetilde{Z}_h$ is the first-draw probability, not generally the inclusion probability.

Let $Y_{-g} = (Y_h)_{h \neq g}$ and let T_{-g} be the Kth-largest perturbed routing score among those other teams. Team g is selected exactly when $Y_g > T_{-g}$. Independence and the Gumbel cumulative distribution give

$$\Pr(g \in \mathcal{S} \mid \mathcal{D}, Y_{-g}) = \rho_g(T_{-g}), \tag{6}$$

$$\rho_g(t) = 1 - \exp[-\exp(\phi_g - t)]\,. \tag{7}$$

For a selected team g, omitting its perturbed routing score leaves exactly $K - 1$ scores above the global $(K+1)$st. Thus T_{-g} equals the observed $(K+1)$st-largest perturbed routing score τ, and the algorithm computes $c_g = \rho_g(\tau)$ for selected teams. This is why selection retains one extra perturbed routing score, not one extra team. The marginal inclusion probability is instead $\pi_g = \mathbb{E}[\rho_g(T_{-g}) \mid \mathcal{D}]$. The correction uses the leave-one-out threshold because the global threshold depends on Y_g.

Unnormalized sums. Let X_g denote either $Z_g = \sum_{i \in \mathcal{T}_g} e^{s_i}$ or one coordinate of $\boldsymbol{U}_g = \sum_{i \in \mathcal{T}_g} e^{s_i} \boldsymbol{v}_i$. These quantities are fixed given $\mathcal{D}$. Let $\mathbf{1}\{g \in \mathcal{S}\}$ denote the indicator that team g is selected. With exact arithmetic,

$$\mathbb{E}\left[\frac{\mathbf{1}\{g \in \mathcal{S}\} X_g}{\rho_g(T_{-g})} \,\middle|\, \mathcal{D}, Y_{-g}\right] = \frac{X_g \,\Pr(g \in \mathcal{S} \mid \mathcal{D}, Y_{-g})}{\rho_g(T_{-g})} = X_g. \tag{8}$$

For selected teams the denominator equals c_g; unselected teams contribute zero. Summing and taking iterated expectations, with the disjoint suffix added exactly, gives

$$\mathbb{E}[\widehat{\boldsymbol{U}} \mid \mathcal{D}] = \boldsymbol{U}, \qquad \mathbb{E}[\widehat{Z} \mid \mathcal{D}] = Z. \tag{9}$$

No factor of $1/K$ is needed. The two sums are unbiased for the current query and cache. Their normalized ratio is generally biased. When $K = M$, all corrections are one and the output equals dense attention on the current cache in exact arithmetic.

Shared team construction. Build teams once per layer and KV head. The parent mean chooses only the first actual-key representative; farthest-first choices then cover keys poorly represented so far. Resolve representative-selection ties by increasing token index and assignment ties by representative-selection order. Forcing each representative to own its row keeps every team nonempty, even when duplicate keys create ties.

Algorithm 2 Build actual-key teams from a parent partition

Require: Prefill keys $(\boldsymbol{k}_i^{\mathrm{pre}})_{i=1}^N$; nonempty parents $\mathcal{P}$ partitioning $\{1, \dots, N\}$; maximum representatives per parent $R \geq 1$.
Ensure: Team records $(\mathcal{T}_g, \boldsymbol{\ell}_g)_{g=1}^M$ partitioning all prompt indices.
$\mathcal{T} \leftarrow$ empty list
for each parent $C \in \mathcal{P}$ **do**
  $r \leftarrow \min(R, |C|)$ and $\boldsymbol{\mu} \leftarrow \dfrac{1}{|C|} \displaystyle\sum_{i \in C} \boldsymbol{k}_i^{\mathrm{pre}}$
  $j_1 \leftarrow \arg\min_{i \in C} \|\boldsymbol{k}_i^{\mathrm{pre}} - \boldsymbol{\mu}\|_2^2$; $L \leftarrow (j_1)$
  for $u = 2, \dots, r$ **do**
    $j_u \leftarrow \arg\max_{i \in C \setminus \{j_1, \dots, j_{u-1}\}} \min_{j \in L} \|\boldsymbol{k}_i^{\mathrm{pre}} - \boldsymbol{k}_j^{\mathrm{pre}}\|_2^2$
    Append j_u to L
  end for
  for each $i \in C$ **do**
    $b_i \leftarrow \arg\min_{u \in \{1, \dots, r\}} \|\boldsymbol{k}_i^{\mathrm{pre}} - \boldsymbol{k}_{j_u}^{\mathrm{pre}}\|_2^2$
  end for
  $b_{j_u} \leftarrow u$ for $u = 1, \dots, r$ ▷ Representative self-ownership
  for $u = 1, \dots, r$ **do**
    Append $\big(\{i \in C : b_i = u\}, \boldsymbol{k}_{j_u}^{\mathrm{pre}}\big)$ to $\mathcal{T}$
  end for
end for
return $\mathcal{T}$

Algorithm 3 Global minibatch k-means parent partition

Require: Prefill keys $(\boldsymbol{k}_i^{\mathrm{pre}})_{i=1}^N$, $N \geq 2$; nominal parent size $P \geq 1$; minibatch k-means settings θ and seed ξ.
Ensure: Nonempty parents $\mathcal{P}$ partitioning all prompt indices.
$M_{\mathrm{parent}} \leftarrow \min\big(N, \max(2, \lfloor N/P \rfloor)\big)$
$(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{M_{\mathrm{parent}}}) \leftarrow \textsc{MiniBatchKMeans}((\boldsymbol{k}_i^{\mathrm{pre}})_{i=1}^N, M_{\mathrm{parent}}; \theta, \xi)$
$b_i \leftarrow \arg\min_{u \in \{1, \dots, M_{\mathrm{parent}}\}} \|\boldsymbol{k}_i^{\mathrm{pre}} - \boldsymbol{\mu}_u\|_2^2$ for every $i = 1, \dots, N$
return $\mathcal{P} = \big\{\{i : b_i = u\} : u = 1, \dots, M_{\mathrm{parent}},\ \{i : b_i = u\} \neq \varnothing\big\}$

Algorithm 4 Contiguous parent partition

Require: Prompt length $N \geq 1$ and parent size $P \geq 1$.
Ensure: Nonempty parents $\mathcal{P}$ partitioning all prompt indices in token order.
$M_{\mathrm{parent}} \leftarrow \lceil N/P \rceil$
$C_u \leftarrow \{(u-1)P + 1, \dots, \min(uP, N)\}$ for $u = 1, \dots, M_{\mathrm{parent}}$
return $\mathcal{P} = \{C_1, \dots, C_{M_{\mathrm{parent}}}\}$

MINIBATCHKMEANS fits squared-Euclidean centers to raw post-RoPE keys. It uses greedy k-means++ initialization, minibatches sampled with replacement, and cumulative-count center updates. For the reported runs, θ uses a minibatch size of 4096, one initialization, at most 100 iterations, a 10-minibatch no-improvement stopping rule, and a reassignment ratio of 0.01. The center-change tolerance is zero and the initialization sample size follows the implementation default. The random seed is $\xi = 0$. Final assignments break ties by center index. Empty parents are omitted. Neither parent construction reorders the original KV cache. The GPU benchmark instead creates a separate packed copy during preparation (Appendix D).

B DIAGNOSTICS AND ABLATIONS

B.1 ADDITIONAL JENSEN EXAMPLES

These examples ask how the distribution of logits within a group changes the error in its mean-key mass surrogate. An arithmetic-mean key gives the mean logit, whereas attention mass sums the exponentials of individual logits. For n keys, let $\bar{\boldsymbol{k}} = n^{-1} \sum_{i=1}^{n} \boldsymbol{k}_i$. Here only, absorb the scale $1/\sqrt{d}$ into the fixed query $\boldsymbol{q}$. Convexity of the exponential gives

$$\exp\left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{q}^\top \boldsymbol{k}_i\right) = \exp\left(\boldsymbol{q}^\top \bar{\boldsymbol{k}}\right) \le \frac{1}{n}\sum_{i=1}^{n} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i\right),$$

$$\implies \quad \sum_{i=1}^{n} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i\right) \ge n \exp\left(\boldsymbol{q}^\top \bar{\boldsymbol{k}}\right). \tag{10}$$

The centroid mass surrogate on the right equals the exact unnormalized mass on the left only when all member logits are equal. Figure 3 holds the group size at 16 logits while changing their distribution. The near-uniform example has the smallest gap. In the bimodal, heavy-tailed, and sparse examples, high-scoring members contribute much more to the exponential sum than to the mean logit.

The different gaps explain why one mean key can distort the relative weights of groups with different logit distributions. The inequality concerns unnormalized mass, and it does not say that every normalized group probability is underestimated. These are constructed examples of the approximation.

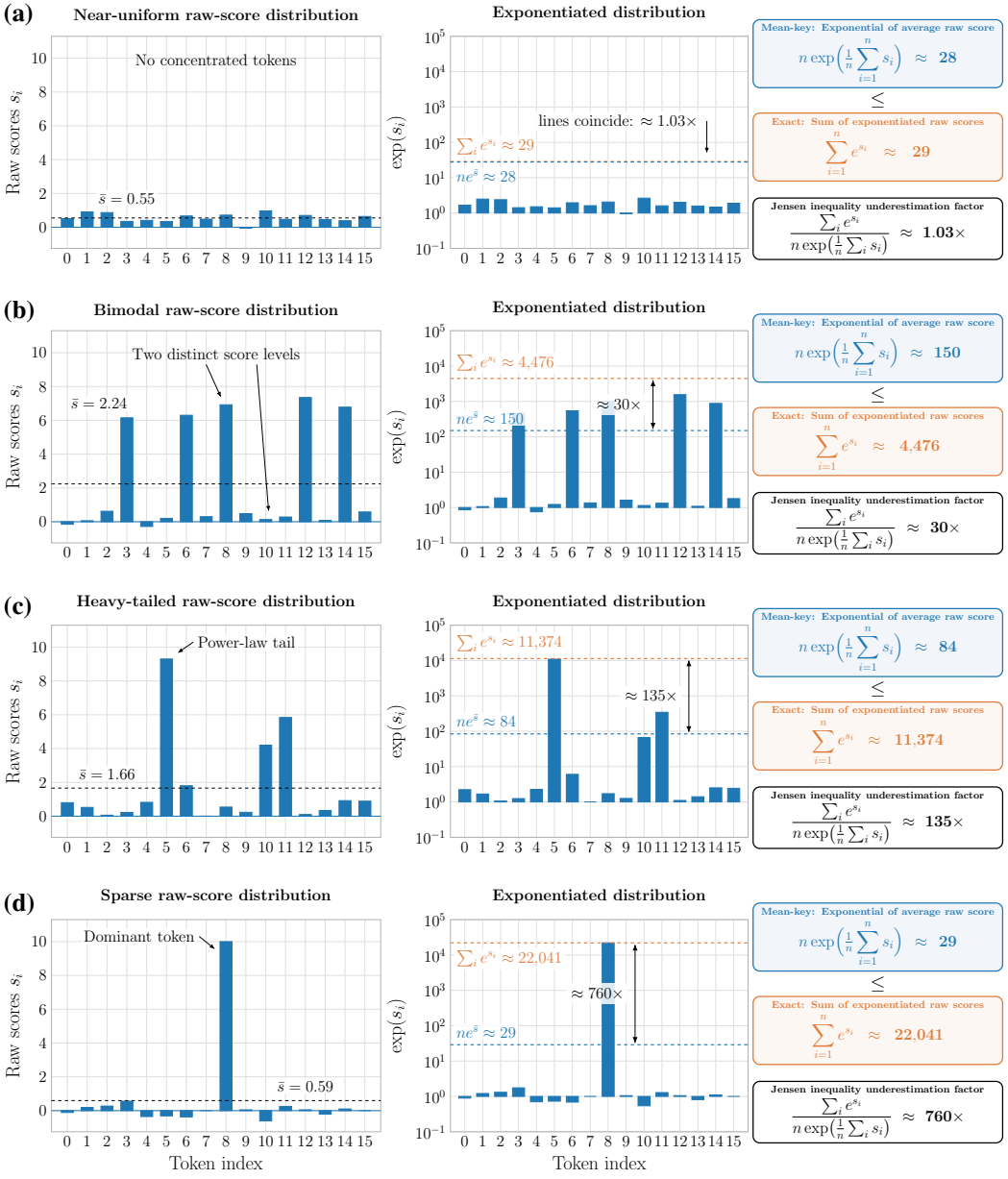


Figure 3: **Additional Jensen examples.** Each row compares 16 raw logits s_i, their exponentials, and the exact-to-surrogate mass ratio $\sum_i e^{s_i} / \left[n \exp(n^{-1} \sum_i s_i)\right]$. The displayed ratios are approximately **(a)** 1.03 for near-uniform logits, **(b)** 30 for bimodal logits, **(c)** 135 for the heavy-tailed example, and **(d)** 760 for the sparse example. The exponentiated-score axes are logarithmic.

B.2 DENSE-TRAJECTORY KL AND TASK-VARIANT EVALUATION

The KL diagnostic asks how the variants compare when their queries and caches are held fixed. For Table 1, we run dense SDPA once per prompt and record its queries and KV states at each measured decode step. Evaluating every variant on these same states removes differences from divergent generated histories. The model is Qwen2.5-7B-Instruct at 8,192-token context, with 20 prompts per task. The mean measured decode-step counts per prompt are 32 for CWE, 9.75 for S-NIAH-1, and 10.55 for QA-1.

Compared distributions. For each recorded query, we compare two distributions over the N tokens in the fixed prompt prefix. The reference probability p_i is the dense SDPA attention weight for token i, restricted to that prefix and renormalized to sum to one. Generated suffix tokens are excluded from this comparison.

For a group g with n_g members, we form its routing probability $P(g)$ by applying a softmax over groups to $\log n_g$ plus the scaled query–key score of its routing key. The team variant uses an actual representative key. The centroid-guided variants use the arithmetic mean of the group's keys. We then spread each group's probability equally across its members. Writing $g(i)$ for the group containing token i, the token proposal and reported divergence are

$$q_i = \frac{P(g(i))}{n_{g(i)}}, \qquad \mathrm{KL}(p\|q) = \sum_{i=1}^{N} p_i \log \frac{p_i}{q_i}. \tag{11}$$

The dense reference is the first argument, and the logarithm is natural, so KL divergence is reported in nats. These probabilities describe the routing proposal. Each grouping covers every prompt token once and has no empty groups. For finite scores, the softmax therefore gives positive probability to every prompt token in exact arithmetic.

Averaging. No sampling is performed in this diagnostic, so the sample budget S does not enter the comparison. Once the query, cached keys, and grouping are fixed, both distributions are fixed. We average KL uniformly over decode steps, all 28 layers, and all 28 query heads within each prompt, then average the prompt means equally within each task. Longer generations therefore do not receive more prompt-level weight. Because the variants differ in both grouping and routing keys, the comparison does not isolate the effect of representative choice alone.

Task-accuracy comparison. The separate task evaluation measures accuracy and logical KV access. Table 6 uses 100 prompts per task across all 13 RULER tasks, with the same model and context length as the KL diagnostic. It compares the $P = 16, R = 4$ team variant with k-means parents against centroid-guided variants at $B = 4$ and $B = 16$, using nominal budgets $S \in \{128, 256, 512\}$.

For the centroid-guided variants, B is the nominal cluster size, not a fixed number of members. With N prompt tokens, it sets the requested number of k-means clusters to $\min\{N, \max(2, \lfloor N/B \rfloor)\}$ per layer and KV head. Clustering uses post-RoPE keys, and actual cluster sizes vary. The $B = 16$ variant matches the team variant's nominal parent size; $B = 4$ provides a finer grouping.

Each centroid-guided variant scores the cluster's mean key and adds the logarithm of its size to form the routing logit. At budget S, it selects $\lfloor S/B \rfloor$ clusters using the same Gumbel sampling without replacement as the team variant. Every member of a selected cluster is read and evaluated with its exact token score. The cluster's attention mass and weighted value sum are divided by its conditional inclusion probability before the corrected sums are combined and normalized.

Whole-team sampling has the highest all-task point estimate at every tested budget, at the cost of the highest reported KV access. At $S = 128$, it reaches 84.16% accuracy compared with 87.09% for SDPA while accessing 24.97% of dense KV. Increasing the team budget narrows the aggregate score gap but requires more access. Both guided variants use fewer logical reads at the same nominal S.

The aggregate ordering hides task-specific trade-offs. Guided $B = 4$ has the higher QA-2 point estimate at every budget, whereas teams have higher point estimates on CWE and VT. Guided $B = 16$ gives the lowest access but has a particularly low MK-NIAH-3 score at $S = 128$ and low VT scores throughout. More sampling does not improve every team result: the VT point estimate falls at $S = 256$ before recovering at $S = 512$, and QA-2 declines between those two budgets. These task differences matter when choosing a budget even though the all-task score increases.

Logical KV access. The reported total includes both routing reads and the keys and values used for attention. Within each KV head, we take the union of prompt rows selected by its seven query heads before counting one key and one value per row. We also count the keys and values in the exactly attended generated suffix. Routing adds one key-vector read per group and KV head at each decode step; it does not read values.

We sum the underlying access counts across decode steps, layers, and prompts, then divide by the corresponding summed dense counts. The reported percentage is therefore a ratio of total counts.

		S-NIAH-1		S-NIAH-2		S-NIAH-3		MK-NIAH-1		MK-NIAH-2	
Method	S	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)
SDPA	—	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	98.00 ± 2.50
SANTA++ (Teams)	128	22.54	100.00 ± 0.00	25.37	100.00 ± 0.00	25.50	100.00 ± 0.00	24.95	99.00 ± 1.50	25.57	94.00 ± 4.50
	256	30.33	100.00 ± 0.00	34.75	99.00 ± 1.50	34.77	100.00 ± 0.00	34.05	100.00 ± 0.00	34.60	98.00 ± 2.50
	512	42.60	100.00 ± 0.00	48.49	100.00 ± 0.00	48.19	100.00 ± 0.00	47.63	100.00 ± 0.00	47.78	95.00 ± 4.50
Centroid-guided ($B = 4$)	128	20.90	100.00 ± 0.00	22.06	100.00 ± 0.00	22.53	98.00 ± 2.50	21.98	99.00 ± 1.50	22.87	98.00 ± 2.50
	256	28.29	100.00 ± 0.00	30.32	100.00 ± 0.00	30.52	98.00 ± 2.50	29.85	100.00 ± 0.00	31.14	97.00 ± 3.00
	512	40.20	100.00 ± 0.00	43.33	100.00 ± 0.00	43.45	100.00 ± 0.00	42.67	98.00 ± 2.50	43.90	96.00 ± 3.50
Centroid-guided ($B = 16$)	128	10.75	97.00 ± 3.50	11.96	97.00 ± 3.50	12.11	87.00 ± 6.50	11.55	96.00 ± 3.50	12.69	81.00 ± 7.50
	256	17.68	99.00 ± 1.50	19.54	98.00 ± 2.50	19.61	97.00 ± 3.01	19.01	98.00 ± 2.50	20.71	92.00 ± 5.00
	512	29.09	100.00 ± 0.00	32.38	100.00 ± 0.00	32.26	100.00 ± 0.00	31.36	100.00 ± 0.00	33.26	94.00 ± 4.50

		MK-NIAH-3		MQ-NIAH		MV-NIAH		VT	
Method	S	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)
SDPA	—	100.00	97.00 ± 3.50	100.00	99.75 ± 0.38	100.00	98.00 ± 1.26	100.00	35.80 ± 9.10
SANTA++ (Teams)	128	23.79	91.00 ± 5.50	25.61	98.25 ± 1.38	25.82	93.50 ± 2.63	22.11	33.80 ± 9.40
	256	31.67	95.00 ± 4.00	34.53	99.75 ± 0.38	34.85	94.25 ± 2.25	29.14	26.00 ± 8.50
	512	43.99	98.00 ± 2.50	47.96	100.00 ± 0.00	48.35	95.75 ± 1.88	40.38	34.80 ± 9.00
Centroid-guided ($B = 4$)	128	21.99	93.00 ± 4.50	22.51	98.50 ± 1.12	22.79	88.00 ± 2.50	20.38	16.20 ± 7.20
	256	29.26	98.00 ± 2.50	30.35	98.00 ± 1.62	30.73	91.00 ± 2.25	26.87	16.40 ± 7.20
	512	41.08	98.00 ± 2.50	43.12	99.25 ± 0.88	43.58	94.50 ± 2.12	37.80	24.80 ± 8.20
Centroid-guided ($B = 16$)	128	12.10	21.00 ± 7.50	12.08	96.75 ± 1.62	12.21	84.50 ± 2.75	10.33	15.20 ± 6.60
	256	19.00	68.00 ± 9.00	19.36	97.75 ± 1.38	19.67	88.00 ± 2.75	16.34	7.80 ± 5.00
	512	30.43	95.00 ± 4.00	31.82	98.75 ± 1.00	32.17	88.75 ± 2.62	26.77	18.80 ± 7.10

		CWE		FWE		QA-1		QA-2		All Tasks	
Method	S	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)	KV (%)	Acc (%)
SDPA	—	100.00	93.30 ± 1.75	100.00	80.33 ± 3.34	100.00	71.00 ± 9.00	100.00	59.00 ± 9.00	100.00	87.09 ± 1.30
SANTA++ (Teams)	128	24.85	88.90 ± 2.30	23.61	78.67 ± 3.84	28.37	65.00 ± 9.00	26.59	52.00 ± 9.50	24.97	84.16 ± 1.45
	256	33.37	89.20 ± 2.50	32.08	79.00 ± 3.84	39.06	67.00 ± 9.50	36.09	57.00 ± 9.50	33.79	84.94 ± 1.34
	512	46.13	92.10 ± 1.80	45.30	80.33 ± 3.17	53.86	70.00 ± 9.00	50.05	55.00 ± 9.50	46.98	86.23 ± 1.34
Centroid-guided ($B = 4$)	128	22.65	82.30 ± 3.10	21.65	77.00 ± 3.18	24.60	65.00 ± 9.50	22.94	55.00 ± 9.50	22.30	82.31 ± 1.35
	256	30.61	85.90 ± 2.30	29.39	79.67 ± 3.66	33.90	72.00 ± 8.50	31.32	59.00 ± 9.50	30.20	84.23 ± 1.27
	512	43.15	90.40 ± 2.25	42.13	83.67 ± 3.50	48.33	70.00 ± 9.01	44.58	59.00 ± 9.50	42.87	85.66 ± 1.32
Centroid-guided ($B = 16$)	128	12.21	66.00 ± 3.55	11.37	70.67 ± 4.51	14.31	69.00 ± 9.00	12.63	53.00 ± 9.50	12.02	71.86 ± 1.66
	256	19.33	77.60 ± 3.10	18.70	74.67 ± 3.83	23.20	67.00 ± 9.00	20.56	53.00 ± 9.01	19.44	78.29 ± 1.49
	512	31.23	80.90 ± 3.60	31.24	79.33 ± 3.84	37.25	69.00 ± 9.00	33.92	55.00 ± 10.00	31.78	83.04 ± 1.34

Table 6: **Task-variant evaluation on RULER at 8K context.** Qwen2.5-7B-Instruct with 100 prompts per task. SANTA++ (teams) uses whole-team sampling with $P = 16, R = 4$; Centroid-guided ($B = 4$) and ($B = 16$) denote the two centroid-guided baselines. S is the nominal budget. KV is the reported theoretical GQA-aware access relative to dense SDPA. Accuracy is reported as mean ± 95% bootstrap confidence interval half-width. *All Tasks* accuracy is the equal-weight mean over all 13 tasks, with a stratified bootstrap that resamples prompts within each task. This is a separate evaluation from the 20-prompt dense-trajectory KL diagnostic.

B.3 CENTROID-DISTANCE DIAGNOSTIC

Contiguous parents avoid clustering, but do they still group nearby keys? Figure 4 compares key clusters, contiguous chunks, and random chunks using the mean Euclidean distance from each raw key vector to the arithmetic centroid of its group. Here, distances to group centroids measure group compactness. The experiment uses Qwen2.5-7B-Instruct at 8,192-token context, with 100 prompts per RULER task and group size $P = 16$.

Key clustering applies k-means independently for each layer and KV head. Contiguous grouping partitions the token sequence into consecutive groups of 16, while the random baseline uses one fixed random shuffle of token positions before partitioning them into groups of 16. For each task, we report the arithmetic mean of token-to-centroid distances over all tokens, KV heads, layers, and prompts.

Across all 13 tasks, contiguous chunks have larger mean distances than key clusters but smaller distances than random chunks. Position thus retains some local structure in the keys without requiring key-space clustering, while key clustering produces the more compact groups on these inputs. This gives a geometric reason to consider contiguous parents rather than random grouping.

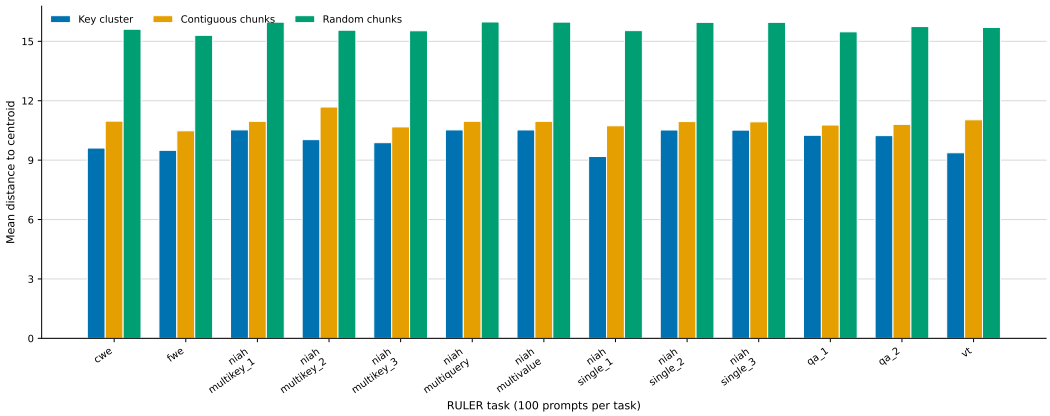


Figure 4: **Key-to-centroid distance across RULER tasks.** Mean distance for key clusters, contiguous chunks, and random chunks on 100 prompts per task, using Qwen2.5-7B-Instruct at 8,192-token context and nominal group size $P = 16$. Contiguous chunks lie between key clusters and random chunks on every plotted task.

B.4 Oracle prefill-window diagnostic

We measure how much attention mass a small number of contiguous windows can retain compared with selecting the same number of tokens individually. This oracle analysis uses dense attention weights from the final prefill query token. It uses 10 prompts per task for CWE and MV-NIAH, with Qwen2.5-7B-Instruct at context lengths of 8,192 and 32,768 tokens. We partition the prompt into disjoint windows of width $w = 16$ and select $K_{\mathrm{w}} = 32$ windows.

For each prompt, layer, and attention head, let a_i be the dense attention weight on token i. Let $\mathcal{I}_{K_{\mathrm{w}}}$ index the K_{w} windows with the largest summed attention weight, and let $\mathcal{W}_j$ contain the token positions in window j. As a reference with the same $wK_{\mathrm{w}} = 512$-token budget, let $\mathcal{D}_{wK_{\mathrm{w}}}$ contain the 512 token positions with the largest individual attention weights for the same prompt, layer, and head. The window effectiveness ratio (WER) is

$$\mathrm{WER} = \frac{\sum_{j \in \mathcal{I}_{K_{\mathrm{w}}}} \sum_{i \in \mathcal{W}_j} a_i}{\sum_{i \in \mathcal{D}_{wK_{\mathrm{w}}}} a_i}. \tag{12}$$

Because the reference set contains the 512 highest-attention tokens, $0 \leq \mathrm{WER} \leq 1$. A WER close to one means that the selected windows retain nearly as much attention mass as the unconstrained 512-token reference set. Table 7 shows that a majority of heads reach the 0.50 threshold in every task–context pair, whereas only 15.1%–23.6% reach 0.90. Thus the 32 windows capture at least half of the reference mass for most heads, while near-complete recovery is less common.

Task	Context length	WER threshold	Heads (%)
CWE	8192	≥ 0.50	82.1 [79.8, 84.9]
CWE	32768	≥ 0.50	80.2 [76.9, 84.8]
CWE	8192	≥ 0.75	54.0 [48.4, 60.7]
CWE	32768	≥ 0.75	49.5 [41.7, 60.7]
CWE	8192	≥ 0.90	22.1 [18.4, 27.0]
CWE	32768	≥ 0.90	23.6 [16.2, 36.4]
MV-NIAH	8192	≥ 0.50	79.0 [75.2, 83.4]
MV-NIAH	32768	≥ 0.50	73.8 [72.0, 75.7]
MV-NIAH	8192	≥ 0.75	51.3 [43.7, 60.7]
MV-NIAH	32768	≥ 0.75	42.6 [39.4, 46.0]
MV-NIAH	8192	≥ 0.90	22.0 [15.6, 30.8]
MV-NIAH	32768	≥ 0.90	15.1 [13.4, 17.0]

Table 7: **Oracle prefill-window effectiveness.** Percentage of heads across all layers meeting each WER threshold from Equation 12. For each prompt, layer, and head, WER compares the attention mass in the top $K_{\mathrm{w}} = 32$ disjoint windows of width $w = 16$ with the mass in the $wK_{\mathrm{w}} = 512$ highest-attention tokens. Results use 10 prompts per task at 8K and 32K context. Brackets give 95% bootstrap confidence intervals from resampling prompts.

B.5 KEY VERSUS PROBE-FINGERPRINT CLUSTERING

Clustering keys directly avoids forming their attention-score fingerprints under probe queries. This separate evaluation uses Qwen2.5-7B-Instruct at 8K context on 50 prompts per task for CWE, MK-NIAH-3, QA-1, and QA-2, with $S \in \{128, 256, 512\}$ and $P = 16, R = 4$.

For key clustering, we apply k-means directly to the unnormalized key vectors independently for each prompt, layer, and KV head. For probe-fingerprint clustering, the final 64 prefill-token queries serve as probes. Each key is represented by its dot-product scores with these probe queries from the query heads sharing its KV head, and each fingerprint dimension is z-score standardized across prompt tokens before k-means. The two variants otherwise use the same team construction, attention estimator, and sampling procedure. Error bars are 95% percentile-bootstrap confidence intervals from 10,000 resamples.

Key clustering has higher point estimates on MK-NIAH-3 at all three budgets, whereas probe fingerprints have higher CWE point estimates at $S = 128$ and 256. On QA-2, the ordering reverses with budget, and both clustering variants remain below the dense point estimate. The QA intervals overlap substantially.

There is no uniform accuracy advantage from constructing probe fingerprints in this comparison. Direct key clustering is a simpler preparation choice, while the lower-budget CWE results identify a setting in which probe fingerprints have the higher point estimates.

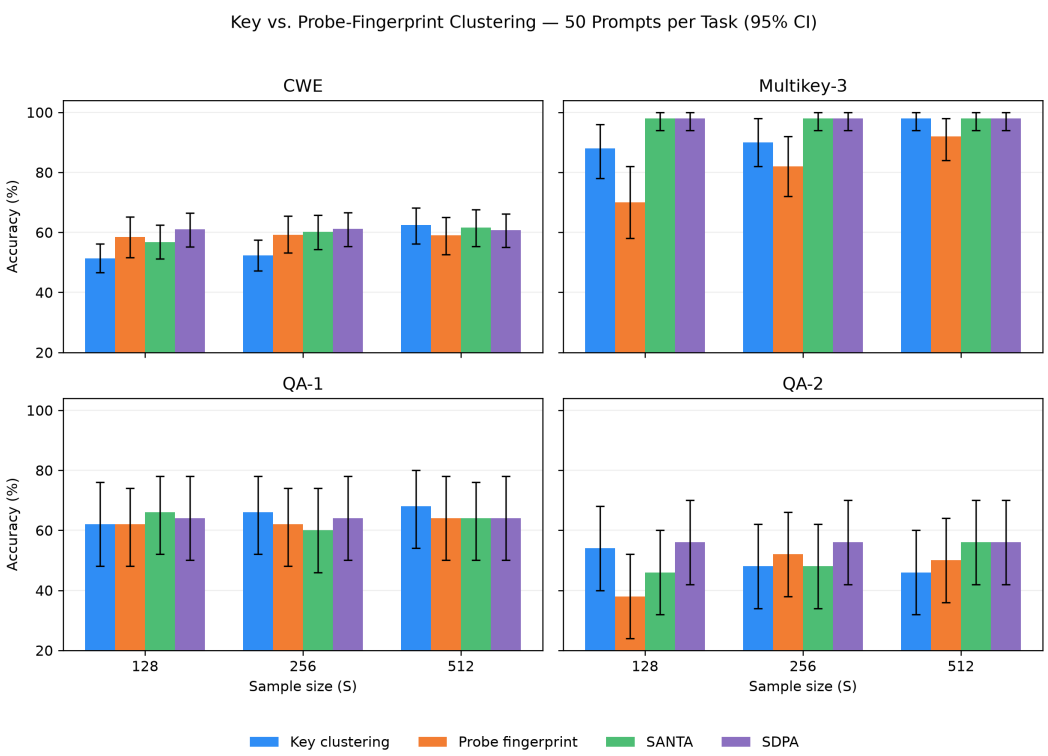


Figure 5: **Key versus probe-fingerprint clustering.** Accuracy on four RULER tasks with 50 prompts per task and nominal budgets $S = 128, 256, 512$. Bars compare key clustering, probe-fingerprint clustering, SANTA, and dense SDPA; error bars denote the displayed 95% confidence intervals.

C Evaluation details and full benchmark results

C.1 Setup and metrics

We evaluate Qwen2.5-7B-Instruct with dense scaled dot-product attention (SDPA), SANTA, and SANTA++ using k-means or contiguous parent groups. HELMET and RULER use 8K and 32K contexts; LongBench v2 uses a 32K context cap, retaining shorter prompts at their original length. Budgets double from $S = 128$ through 2048 for HELMET RAG, through 1024 for LongBench v2 and 8K RULER, and through 4096 for 32K RULER. The complete 32K aggregate sweeps are reported in Tables 2, 3, and 4, respectively.

For SANTA++, the nominal budget requests $K = \min\{M, S/(P/R)\}$ teams, with integral P/R and $S/(P/R)$, where M is the number of teams. This conversion requires the run's P, R; it does not give the number of unique rows read. As specified in Appendix A, the reference partitions the prompt after dense prefill and keeps newly generated cache rows in a separate exact suffix. That suffix starts empty and grows as generated tokens are fed back; no prompt window is reserved for exact evaluation.

Absolute scores retain each benchmark's metric. The full aggregate tables retain their normalized columns, computed as the absolute score divided by the corresponding SDPA score from the same benchmark and context, expressed as a percentage. HELMET, LongBench, and RULER report 95% bootstrap confidence interval half-widths.

Reference logical-access counter. For one layer, KV head, and decoding call c, let N_c be the current cache length, e_c the exact generated-suffix length, and m_c the number of team representatives scanned for routing. Let $I_{c,h}$ be the selected prompt-row indices for query head h and set $u_c = |\bigcup_h I_{c,h}|$, with the union restricted to query heads sharing that KV head. The reference counts one

key and one value per dense row and computes

$$\mathrm{KV}_{\mathrm{total}}(\%) = 100\,\frac{\sum_c \big[m_c + 2(u_c + e_c)\big]}{\sum_c 2N_c}. \tag{13}$$

The selected prompt rows and exact suffix are disjoint. A representative read for routing and again as a selected member counts in both passes. Counts also add across layers and decoding steps. Prefill and team construction are outside this counter. For SANTA, the corresponding numerator per call is N_c plus the number of distinct sampled value rows across sharing query heads, so its full-key scan alone costs 50%.

Access aggregation. The reference also records a selected-KV-only access measure that omits the routing term m_c. The main RULER tables instead use the total-access measure in Equation 13, which includes routing. For RULER aggregates, we sum the raw numerators and denominators across prompts and tasks before division rather than averaging the displayed task percentages. For HELMET RAG, aggregate access is the equal-weight mean of the four RAG task-level total-access percentages, including representative-key routing reads (Appendix C.2).

C.2 HELMET

Table 2 reports the main 32K RAG sweep, and Table 8 gives the corresponding 8K sweep. Both use $S \in \{128, 256, 512, 1024, 2048\}$. The RAG score is the equal-weight mean of substring exact match on Natural Questions, TriviaQA, PopQA, and HotpotQA. KV access is the equal-weight mean of those four tasks' total-access percentages, including representative-key routing reads. Tables 9 and 10 give individual RAG task results at all five budgets; the non-RAG tasks and the all-task summary cover $S \in \{128, 256, 512\}$ only.

The additional tasks expose losses outside the RAG mean. At 32K, JSON KV scores fall from SDPA's 96.80% to 45.60% with k-means and 33.40% with contiguous grouping at $S = 128$. At $S = 512$, these rise to 87.00% and 68.60%, respectively. Non-RAG tasks use their own scoring metrics. The detailed tables' "All tasks average" summarizes all 13 tasks.

Method	S	KV (%)	Acc (norm) (%)	Acc (abs) (%)
SDPA	–	100.00	100.00	62.65 ± 2.02
SANTA	128	51.62	100.24	62.80 ± 2.02
	256	52.70	99.60	62.40 ± 2.03
	512	54.39	99.60	62.40 ± 2.03
	1024	56.77	100.16	62.75 ± 2.05
	2048	60.14	100.00	62.65 ± 2.07
SANTA++ (k-means)	128	27.55	97.85	61.30 ± 1.98
	256	37.60	98.72	61.85 ± 2.00
	512	51.83	99.28	62.20 ± 2.00
	1024	69.95	98.56	61.75 ± 2.00
	2048	90.18	100.40	62.90 ± 2.05
SANTA++ (Contiguous)	128	23.69	96.09	60.20 ± 2.02
	256	32.38	100.80	63.15 ± 2.03
	512	45.73	99.44	62.30 ± 2.02
	1024	64.10	99.52	62.35 ± 2.05
	2048	86.28	99.76	62.50 ± 2.07

Table 8: **HELMET RAG: full 8K budget sweep with Qwen2.5-7B-Instruct.** Acc (abs) is the equal-weight mean substring exact match on Natural Questions, TriviaQA, PopQA, and HotpotQA. Acc (norm) is the absolute score divided by the corresponding SDPA score at the same context length, expressed as a percentage. KV is the equal-weight mean of the four RAG task-level total-access percentages, including representative-key routing reads. S is the nominal budget. The ± entries are the reported score-interval half-widths.

(a) RAG and reranking

Method	S	Natural Questions		TriviaQA		PopQA		HotpotQA		MS MARCO	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	53.20 ± 4.30	100.00	82.20 ± 3.30	100.00	64.40 ± 4.10	100.00	50.80 ± 4.40	100.00	58.51 ± 3.89
SANTA	128	51.62	54.40 ± 4.40	51.55	82.40 ± 3.40	51.79	63.00 ± 4.20	51.51	51.40 ± 4.30	51.72	59.16 ± 3.81
	256	52.73	53.80 ± 4.40	52.58	81.80 ± 3.40	52.97	63.80 ± 4.20	52.52	50.20 ± 4.50	52.75	58.03 ± 3.75
	512	54.46	53.00 ± 4.30	54.21	82.20 ± 3.30	54.79	64.20 ± 4.20	54.09	50.20 ± 4.50	54.27	59.06 ± 3.84
	1024	56.90	54.40 ± 4.30	56.49	82.20 ± 3.30	57.30	64.00 ± 4.10	56.39	50.40 ± 4.40		
	2048	60.37	54.00 ± 4.30	59.77	82.20 ± 3.30	60.83	64.00 ± 4.20	59.58	50.40 ± 4.40		
SANTA++ (K-means)	128	27.49	53.80 ± 4.40	26.87	83.20 ± 3.30	28.88	61.60 ± 4.30	26.96	46.60 ± 4.40	28.23	56.99 ± 3.84
	256	37.46	54.80 ± 4.30	36.64	81.20 ± 3.50	39.55	64.20 ± 4.30	36.73	47.20 ± 4.30	37.32	57.91 ± 3.91
	512	51.58	53.20 ± 4.50	50.57	83.20 ± 3.30	54.46	63.00 ± 4.20	50.69	49.40 ± 4.40	50.36	59.02 ± 3.67
	1024	69.43	52.40 ± 4.40	68.53	81.80 ± 3.30	73.13	63.00 ± 4.20	68.70	49.80 ± 4.30		
	2048	89.46	54.20 ± 4.30	88.82	82.80 ± 3.30	93.44	64.00 ± 4.20	88.98	50.60 ± 4.40		
SANTA++ (Contiguous)	128	23.66	53.00 ± 4.40	23.05	80.40 ± 3.40	24.80	60.80 ± 4.30	23.25	46.60 ± 4.30	25.03	44.95 ± 3.99
	256	32.28	55.20 ± 4.30	31.41	82.20 ± 3.40	34.15	63.80 ± 4.30	31.69	51.40 ± 4.40	33.13	49.59 ± 4.04
	512	45.51	53.60 ± 4.40	44.35	82.40 ± 3.30	48.34	63.00 ± 4.40	44.72	50.20 ± 4.40	45.37	54.81 ± 3.53
	1024	63.72	54.40 ± 4.30	62.44	81.60 ± 3.30	67.18	63.60 ± 4.30	63.07	49.80 ± 4.40		
	2048	85.50	53.20 ± 4.30	84.51	81.80 ± 3.40	90.06	64.20 ± 4.20	85.03	50.80 ± 4.40		

Method	S	TREC coarse		TREC fine		NLU		Banking77		Clinic150	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	83.00 ± 3.20	100.00	41.80 ± 4.40	100.00	79.60 ± 3.50	100.00	75.20 ± 3.80	100.00	81.60 ± 3.40
SANTA	128	52.03	83.20 ± 3.30	52.00	40.20 ± 4.30	52.00	78.00 ± 3.60	52.01	76.00 ± 3.60	51.96	79.40 ± 3.60
	256	53.39	83.00 ± 3.20	53.36	43.20 ± 4.30	53.38	79.60 ± 3.50	53.34	76.20 ± 3.70	53.30	80.40 ± 3.50
	512	55.47	82.80 ± 3.30	55.44	42.40 ± 4.30	55.50	80.00 ± 3.50	55.37	75.60 ± 3.70	55.37	82.20 ± 3.40
SANTA++ (K-means)	128	28.41	79.20 ± 3.60	27.95	39.00 ± 4.20	27.79	77.40 ± 3.60	27.41	74.20 ± 3.90	27.61	74.20 ± 3.90
	256	38.74	83.00 ± 3.40	37.99	41.60 ± 4.30	37.85	78.20 ± 3.50	37.14	76.00 ± 3.70	37.39	77.80 ± 3.70
	512	53.25	83.20 ± 3.20	52.16	41.80 ± 4.40	51.96	79.60 ± 3.60	50.94	76.20 ± 3.70	51.15	79.60 ± 3.50
SANTA++ (Contiguous)	128	23.95	80.40 ± 3.40	23.71	35.80 ± 4.20	23.89	75.20 ± 3.90	23.47	72.80 ± 3.90	23.43	74.00 ± 3.80
	256	32.85	83.40 ± 3.20	32.44	39.20 ± 4.40	32.74	79.20 ± 3.50	31.97	76.60 ± 3.80	31.94	77.20 ± 3.70
	512	46.45	83.20 ± 3.30	45.81	42.80 ± 4.40	46.19	78.00 ± 3.70	44.98	76.00 ± 3.70	44.96	79.00 ± 3.50

Method	S	InfBench QA		InfBench MC		JSON KV		All tasks average	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	20.08 ± 3.54	100.00	40.61 ± 6.33	100.00	99.40 ± 0.70	100.00	63.88 ± 3.76
SANTA	128	51.24	19.22 ± 3.51	50.99	42.36 ± 6.33	51.29	99.40 ± 0.70	51.67	63.70 ± 3.77
	256	52.07	19.95 ± 3.54	51.58	43.23 ± 6.33	52.09	99.40 ± 0.70	52.77	64.05 ± 3.77
	512	53.39	19.83 ± 3.54	52.50	41.48 ± 6.33	53.34	99.60 ± 0.50	54.48	64.04 ± 3.75
SANTA++ (K-means)	128	26.02	18.68 ± 3.48	24.33	43.23 ± 6.34	24.08	89.80 ± 2.70	27.08	61.38 ± 4.00
	256	35.80	20.15 ± 3.50	33.88	42.79 ± 6.55	32.21	98.00 ± 1.30	36.82	63.30 ± 3.87
	512	50.09	20.05 ± 3.62	48.38	43.67 ± 6.12	45.09	99.40 ± 0.70	50.82	63.95 ± 3.76
SANTA++ (Contiguous)	128	22.44	18.52 ± 3.40	21.19	42.36 ± 6.33	22.08	74.20 ± 3.80	23.38	58.39 ± 4.09
	256	30.71	18.51 ± 3.35	29.08	41.48 ± 6.55	29.37	92.20 ± 2.30	31.83	62.31 ± 3.94
	512	43.91	20.26 ± 3.56	42.28	41.92 ± 6.55	41.36	98.20 ± 1.10	44.94	63.34 ± 3.83

Table 9: **HELMET task results at 8K context length.** Each accuracy is reported as the mean ± the half-width of its 95% confidence interval.

Method	S	Natural Questions		TriviaQA		PopQA		HotpotQA		MS MARCO	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	46.20 ± 4.30	100.00	82.00 ± 3.50	100.00	62.80 ± 4.30	100.00	38.60 ± 4.20	100.00	34.39 ± 4.36
SANTA	128	50.43	46.40 ± 4.30	50.41	81.00 ± 3.50	50.46	62.20 ± 4.20	50.41	38.00 ± 4.30	50.46	37.63 ± 4.60
	256	50.77	45.60 ± 4.40	50.71	81.00 ± 3.40	50.81	62.80 ± 4.20	50.71	38.00 ± 4.40	50.77	35.79 ± 4.58
	512	51.33	46.00 ± 4.40	51.23	81.40 ± 3.30	51.40	62.80 ± 4.20	51.23	39.80 ± 4.20	51.29	34.49 ± 4.35
	1024	52.23	46.40 ± 4.40	52.05	80.40 ± 3.40	52.32	62.80 ± 4.20	52.06	38.80 ± 4.10		
	2048	53.65	46.80 ± 4.40	53.35	80.40 ± 3.40	53.76	62.80 ± 4.30	53.37	38.80 ± 4.30		
SANTA++ (K-means)	128	17.48	43.80 ± 4.30	17.27	80.80 ± 3.40	17.86	54.40 ± 4.40	17.34	37.80 ± 4.20	17.40	27.70 ± 4.14
	256	21.80	44.60 ± 4.30	21.50	79.00 ± 3.50	22.37	59.60 ± 4.20	21.60	36.20 ± 4.30	21.34	28.98 ± 4.33
	512	28.56	46.40 ± 4.40	28.12	79.00 ± 3.50	29.39	59.40 ± 4.30	28.25	36.60 ± 4.20	27.55	30.30 ± 4.35
	1024	38.43	45.00 ± 4.30	37.95	81.00 ± 3.40	39.60	61.00 ± 4.30	37.97	38.00 ± 4.40		
	2048	52.02	47.20 ± 4.40	51.41	81.40 ± 3.50	53.65	62.20 ± 4.30	51.44	37.00 ± 4.20		
SANTA++ (Contiguous)	128	15.66	42.80 ± 4.40	15.51	75.60 ± 3.80	15.90	58.60 ± 4.40	15.58	35.40 ± 4.20	15.87	17.56 ± 3.39
	256	18.46	42.60 ± 4.30	18.23	77.20 ± 3.80	18.89	61.00 ± 4.30	18.36	36.80 ± 4.20	18.50	23.52 ± 3.87
	512	23.29	44.60 ± 4.30	22.92	76.20 ± 3.70	24.03	60.20 ± 4.30	23.14	37.20 ± 4.20	23.03	24.69 ± 4.03
	1024	31.09	46.00 ± 4.40	30.66	78.60 ± 3.60	32.27	61.00 ± 4.20	30.88	38.40 ± 4.30		
	2048	43.15	46.40 ± 4.40	42.50	78.00 ± 3.60	44.91	62.00 ± 4.30	42.80	37.00 ± 4.20		

Method	S	TREC coarse		TREC fine		NLU		Banking77		Clinic150	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	83.20 ± 3.30	100.00	50.80 ± 4.40	100.00	86.20 ± 3.00	100.00	83.40 ± 3.30	100.00	89.80 ± 2.60
SANTA	128	50.61	83.20 ± 3.30	50.59	48.80 ± 4.20	50.58	85.00 ± 3.10	50.58	83.20 ± 3.30	50.56	89.00 ± 2.90
	256	51.07	83.60 ± 3.30	51.03	50.80 ± 4.30	51.02	86.20 ± 3.00	51.01	83.20 ± 3.20	51.00	90.80 ± 2.50
	512	51.84	83.60 ± 3.20	51.78	50.80 ± 4.30	51.77	84.60 ± 3.20	51.72	82.40 ± 3.30	51.72	89.60 ± 2.70
SANTA++ (K-means)	128	17.95	81.60 ± 3.50	17.79	46.60 ± 4.50	17.52	80.80 ± 3.40	17.31	83.00 ± 3.30	17.46	86.40 ± 3.00
	256	22.50	84.40 ± 3.10	22.19	47.80 ± 4.40	21.85	85.20 ± 3.20	21.47	83.40 ± 3.30	21.69	89.60 ± 2.60
	512	29.57	85.00 ± 3.20	29.03	50.80 ± 4.40	28.66	84.40 ± 3.10	28.05	81.40 ± 3.40	28.31	88.20 ± 2.80
SANTA++ (Contiguous)	128	15.74	83.80 ± 3.20	15.61	41.60 ± 4.40	15.64	78.40 ± 3.70	15.55	78.60 ± 3.60	15.56	83.60 ± 3.30
	256	18.66	85.60 ± 3.00	18.41	44.00 ± 4.50	18.49	80.40 ± 3.40	18.30	81.00 ± 3.50	18.30	86.40 ± 3.00
	512	23.71	85.80 ± 3.00	23.28	46.20 ± 4.30	23.43	82.60 ± 3.40	23.07	81.60 ± 3.30	23.07	87.80 ± 2.80

Method	S	InfBench QA		InfBench MC		JSON KV		All tasks average	
		Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)	Total access (%)	Acc (%)
SDPA	–	100.00	15.00 ± 2.54	100.00	51.09 ± 6.55	100.00	96.80 ± 1.50	100.00	63.10 ± 3.68
SANTA	128	50.36	13.87 ± 2.36	50.28	46.29 ± 6.55	50.36	96.40 ± 1.70	50.47	62.38 ± 3.72
	256	50.62	14.80 ± 2.55	50.48	48.47 ± 6.55	50.60	96.80 ± 1.50	50.82	62.91 ± 3.68
	512	51.07	14.52 ± 2.51	50.79	48.47 ± 6.55	51.00	96.80 ± 1.50	51.40	62.71 ± 3.67
SANTA++ (K-means)	128	16.97	13.40 ± 2.28	16.42	50.66 ± 6.55	15.87	45.60 ± 4.50	17.28	56.35 ± 3.96
	256	21.14	14.45 ± 2.54	20.51	48.91 ± 6.55	19.01	70.20 ± 4.10	21.46	59.41 ± 3.88
	512	27.80	13.73 ± 2.42	27.14	48.47 ± 6.55	24.35	87.00 ± 3.00	28.06	60.82 ± 3.82
SANTA++ (Contiguous)	128	15.27	11.92 ± 2.05	14.94	46.72 ± 6.55	15.00	33.40 ± 4.10	15.53	52.92 ± 3.93
	256	17.85	12.61 ± 2.27	17.41	48.47 ± 6.55	17.18	54.60 ± 4.40	18.23	56.48 ± 3.93
	512	22.39	13.54 ± 2.30	21.95	48.91 ± 6.55	21.20	68.60 ± 4.00	22.96	58.30 ± 3.86

Table 10: HELMET task results at 32K context length. Each accuracy is reported as the mean ± the half-width of its 95% confidence interval.

C.3 LONGBENCH V2

Table 3 reports the complete aggregate budget sweep on 503 distinct LongBench v2 prompts using Qwen2.5-7B-Instruct. We use the repository's middle-truncation scheme with a 32K context cap. Prompts longer than the cap are middle-truncated. Prompts at or below the cap retain their original length.

Prompting. Each prompt begins with Qwen's assistant preamble. We modify the repository's prompting to request brief reasoning and a fixed answer format by appending the following instruction:

> Keep your reasoning concise (under 3 sentences). There is exactly one correct option. On the final line, provide your answer in the exact format: 'Final Answer: [X]' where X is a single letter (A, B, C, or D).

C.4 RULER

Table 4 reports the full 32K aggregate sweep. Tables 11 and 12 give all 13 tasks: S-NIAH-1/2/3, MK-NIAH-1/2/3, MQ-NIAH, MV-NIAH, VT, CWE, FWE, QA-1, and QA-2. The main RULER evaluations use the reference repository's task definitions, prompting, and grading conventions with Qwen2.5-7B-Instruct. At both 8K and 32K, all 13 tasks have 500 completed prompts per task. Smaller diagnostic and MLA runs are separate and are not pooled into these results.

Each task has its own reported GQA-aware KV percentage. The main RULER results use the total-access measure in Equation 13, which includes both selected GQA KV reads and representative-key reads used for routing. Aggregate access pools the underlying read counts across prompts and tasks before division rather than averaging the displayed task percentages.

Method	S	S-NIAH-1 KV (%)	S-NIAH-1 Acc (%)	S-NIAH-2 KV (%)	S-NIAH-2 Acc (%)	S-NIAH-3 KV (%)	S-NIAH-3 Acc (%)	MK-NIAH-1 KV (%)	MK-NIAH-1 Acc (%)	MK-NIAH-2 KV (%)	MK-NIAH-2 Acc (%)
SDPA	—	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	98.80 ± 0.90
SANTA	128	50.97	100.00 ± 0.00	51.11	100.00 ± 0.00	51.06	99.80 ± 0.30	51.13	100.00 ± 0.00	51.25	98.80 ± 0.90
	256	51.56	100.00 ± 0.00	51.81	100.00 ± 0.00	51.72	99.80 ± 0.30	51.83	100.00 ± 0.00	52.08	99.00 ± 0.90
	512	52.52	100.00 ± 0.00	52.93	100.00 ± 0.00	52.76	100.00 ± 0.00	52.94	100.00 ± 0.00	53.41	98.20 ± 1.10
	1024	53.98	100.00 ± 0.00	54.62	100.00 ± 0.00	54.35	100.00 ± 0.00	54.62	100.00 ± 0.00	55.43	98.20 ± 1.10
SANTA++ (K-means)	128	22.53	100.00 ± 0.00	25.43	99.40 ± 0.80	25.46	98.00 ± 1.20	24.95	98.40 ± 1.10	25.57	96.00 ± 1.60
	256	30.37	100.00 ± 0.00	34.75	99.60 ± 0.50	34.63	99.60 ± 0.50	34.04	99.80 ± 0.30	34.59	98.40 ± 1.10
	512	42.54	100.00 ± 0.00	48.54	100.00 ± 0.00	48.16	100.00 ± 0.00	47.57	100.00 ± 0.00	47.76	97.80 ± 1.20
	1024	60.33	100.00 ± 0.00	66.64	100.00 ± 0.00	66.20	100.00 ± 0.00	65.69	100.00 ± 0.00	65.51	97.60 ± 1.30
SANTA++ (Contiguous)	128	20.10	100.00 ± 0.00	21.99	99.20 ± 0.70	22.46	98.40 ± 1.10	21.78	99.00 ± 0.90	22.59	91.40 ± 2.50
	256	27.08	100.00 ± 0.00	29.86	99.00 ± 0.90	30.13	99.60 ± 0.60	29.38	98.60 ± 1.10	30.71	96.20 ± 1.60
	512	38.72	100.00 ± 0.00	42.36	100.00 ± 0.00	42.49	99.80 ± 0.30	41.68	100.00 ± 0.00	43.34	97.40 ± 1.40
	1024	55.97	100.00 ± 0.00	60.42	100.00 ± 0.00	60.36	99.80 ± 0.40	59.62	100.00 ± 0.00	60.94	98.00 ± 1.30

Method	S	MK-NIAH-3 KV (%)	MK-NIAH-3 Acc (%)	MQ-NIAH KV (%)	MQ-NIAH Acc (%)	MV-NIAH KV (%)	MV-NIAH Acc (%)	VT KV (%)	VT Acc (%)
SDPA	—	100.00	97.00 ± 1.40	100.00	99.90 ± 0.12	100.00	97.75 ± 0.65	100.00	30.60 ± 3.78
SANTA	128	51.19	96.40 ± 1.50	51.19	99.90 ± 0.12	51.21	97.25 ± 0.70	51.05	28.44 ± 3.84
	256	51.94	96.20 ± 1.60	51.90	99.85 ± 0.17	51.96	97.20 ± 0.70	51.63	29.52 ± 3.80
	512	53.10	96.40 ± 1.50	53.00	99.90 ± 0.12	53.12	97.50 ± 0.70	52.50	30.56 ± 3.96
	1024	54.85	96.60 ± 1.50	54.64	99.85 ± 0.17	54.86	97.50 ± 0.65	53.76	31.84 ± 4.04
SANTA++ (K-means)	128	23.80	87.60 ± 2.80	25.59	99.00 ± 0.48	25.78	92.30 ± 1.12	22.11	24.56 ± 3.64
	256	31.71	94.80 ± 1.90	34.51	99.75 ± 0.23	34.86	94.85 ± 0.93	29.16	32.00 ± 4.00
	512	44.02	96.80 ± 1.40	47.92	99.85 ± 0.17	48.33	96.05 ± 0.82	40.39	35.52 ± 4.20
	1024	61.63	96.80 ± 1.40	65.91	99.85 ± 0.17	66.35	96.80 ± 0.73	56.94	30.60 ± 3.94
SANTA++ (Contiguous)	128	21.97	42.40 ± 4.30	22.37	97.80 ± 0.68	22.47	87.50 ± 1.38	19.52	17.12 ± 3.34
	256	29.09	80.40 ± 3.30	29.90	98.65 ± 0.48	30.14	90.65 ± 1.20	25.46	21.80 ± 3.38
	512	40.49	93.00 ± 2.10	42.02	99.70 ± 0.23	42.43	93.35 ± 1.05	35.86	25.32 ± 3.76
	1024	57.42	95.60 ± 1.80	59.76	99.85 ± 0.17	60.26	95.90 ± 0.83	51.86	28.60 ± 3.98

Method	S	CWE KV (%)	CWE Acc (%)	FWE KV (%)	FWE Acc (%)	QA-1 KV (%)	QA-1 Acc (%)	QA-2 KV (%)	QA-2 Acc (%)
SDPA	—	100.00	93.86 ± 0.86	100.00	81.60 ± 1.53	100.00	74.60 ± 3.80	100.00	55.20 ± 4.50
SANTA	128	51.27	91.62 ± 0.99	51.35	82.33 ± 1.57	51.47	75.00 ± 4.00	51.28	54.60 ± 4.30
	256	52.11	93.32 ± 0.86	52.26	81.80 ± 1.53	52.45	74.40 ± 3.80	52.11	55.20 ± 4.30
	512	53.42	93.92 ± 0.89	53.70	81.93 ± 1.53	54.02	74.40 ± 3.80	53.43	55.40 ± 4.50
	1024	55.39	94.04 ± 0.76	55.89	81.53 ± 1.50	56.39	74.40 ± 3.80	55.42	56.20 ± 4.30
SANTA++ (K-means)	128	24.85	89.80 ± 0.90	23.62	80.33 ± 1.53	28.35	72.20 ± 3.80	26.61	52.60 ± 4.50
	256	33.40	92.10 ± 0.95	32.10	81.33 ± 1.63	38.81	74.00 ± 3.90	36.31	52.00 ± 4.30
	512	46.19	92.72 ± 0.94	45.33	81.67 ± 1.47	53.51	73.60 ± 3.80	50.27	53.40 ± 4.40
	1024	63.64	93.16 ± 0.98	63.78	82.13 ± 1.50	72.14	75.20 ± 3.80	68.31	54.60 ± 4.40
SANTA++ (Contiguous)	128	21.95	73.92 ± 1.28	20.99	74.33 ± 1.73	24.45	68.80 ± 4.20	23.15	51.40 ± 4.20
	256	29.34	82.72 ± 1.17	28.40	78.13 ± 1.73	33.65	72.40 ± 3.80	31.60	53.80 ± 4.40
	512	41.31	89.38 ± 0.98	40.66	80.60 ± 1.63	47.67	73.60 ± 3.90	44.56	53.40 ± 4.20
	1024	58.57	90.72 ± 0.95	58.84	81.13 ± 1.50	66.71	74.60 ± 3.80	62.86	54.40 ± 4.40

Table 11: **Per-task accuracy and KV memory access on RULER (8K context) on Qwen2.5-7B-Instruct.** *KV* is the theoretical GQA-aware KV memory access (%), and S denotes the nominal budget. Accuracy values are reported as mean ± 95% bootstrap confidence interval half-width.

Method	S	S-NIAH-1 KV (%)	S-NIAH-1 Acc (%)	S-NIAH-2 KV (%)	S-NIAH-2 Acc (%)	S-NIAH-3 KV (%)	S-NIAH-3 Acc (%)	MK-NIAH-1 KV (%)	MK-NIAH-1 Acc (%)	MK-NIAH-2 KV (%)	MK-NIAH-2 Acc (%)
SDPA	—	100.00	100.00 ± 0.00	100.00	100.00 ± 0.00	100.00	98.60 ± 1.00	100.00	99.00 ± 0.90	100.00	93.40 ± 2.20
SANTA	128	50.28	100.00 ± 0.00	50.31	100.00 ± 0.00	50.30	99.20 ± 0.70	50.31	99.20 ± 0.70	50.37	93.20 ± 2.20
	256	50.47	100.00 ± 0.00	50.53	100.00 ± 0.00	50.51	98.80 ± 0.90	50.53	99.00 ± 0.90	50.64	92.80 ± 2.20
	512	50.81	100.00 ± 0.00	50.91	100.00 ± 0.00	50.87	98.80 ± 0.90	50.90	99.00 ± 0.90	51.12	93.60 ± 2.10
	1024	51.37	100.00 ± 0.00	51.53	100.00 ± 0.00	51.45	98.60 ± 1.00	51.51	99.00 ± 0.90	51.92	93.40 ± 2.10
	2048	52.28	100.00 ± 0.00	52.54	100.00 ± 0.00	52.39	98.60 ± 1.00	52.49	99.00 ± 0.90	53.17	93.40 ± 2.20
	4096	53.69	100.00 ± 0.00	54.03	100.00 ± 0.00	53.82	98.60 ± 1.10	53.97	99.00 ± 0.90	55.02	93.60 ± 2.10
SANTA++ (K-means)	128	15.65	97.40 ± 1.30	16.82	98.80 ± 0.90	16.79	96.40 ± 1.70	16.69	95.40 ± 1.70	16.55	77.00 ± 3.90
	256	18.71	98.80 ± 0.90	20.64	99.40 ± 0.70	20.56	98.60 ± 1.00	20.50	95.80 ± 1.80	20.31	87.00 ± 3.00
	512	23.77	100.00 ± 0.00	26.65	99.60 ± 0.50	26.59	99.00 ± 0.90	26.49	96.60 ± 1.60	26.24	88.20 ± 2.90
	1024	31.77	100.00 ± 0.00	35.71	100.00 ± 0.00	35.62	99.20 ± 0.70	35.51	98.40 ± 1.00	35.17	89.60 ± 2.80
	2048	43.85	100.00 ± 0.00	48.37	100.00 ± 0.00	48.33	99.40 ± 0.70	48.27	99.00 ± 0.90	47.90	92.60 ± 2.30
	4096	61.16	100.00 ± 0.00	64.94	100.00 ± 0.00	64.89	99.20 ± 0.70	64.85	99.00 ± 0.90	64.80	93.40 ± 2.20
SANTA++ (Contiguous)	128	14.63	99.40 ± 0.70	15.08	91.40 ± 2.40	15.16	86.80 ± 3.00	15.01	90.20 ± 2.60	15.19	47.40 ± 4.80
	256	16.74	99.80 ± 0.30	17.42	96.60 ± 1.60	17.53	94.00 ± 2.00	17.33	94.40 ± 1.90	17.72	65.20 ± 4.20
	512	20.66	99.80 ± 0.30	21.55	98.20 ± 1.10	21.69	97.60 ± 1.30	21.42	97.00 ± 1.50	22.15	80.20 ± 3.30
	1024	27.40	100.00 ± 0.00	28.47	99.60 ± 0.50	28.63	98.80 ± 1.10	28.32	98.60 ± 0.90	29.43	88.20 ± 2.80
	2048	38.21	100.00 ± 0.00	39.36	100.00 ± 0.00	39.55	99.00 ± 0.90	39.17	99.00 ± 0.90	40.66	91.60 ± 2.40
	4096	54.15	100.00 ± 0.00	55.24	100.00 ± 0.00	55.45	99.60 ± 0.50	55.14	99.40 ± 0.70	56.81	93.20 ± 2.20

Method	S	MK-NIAH-3 KV (%)	MK-NIAH-3 Acc (%)	MQ-NIAH KV (%)	MQ-NIAH Acc (%)	MV-NIAH KV (%)	MV-NIAH Acc (%)	VT KV (%)	VT Acc (%)
SDPA	—	100.00	89.60 ± 2.70	100.00	99.40 ± 0.33	100.00	90.35 ± 1.17	100.00	33.32 ± 3.58
SANTA	128	50.33	88.40 ± 2.80	50.31	99.25 ± 0.38	50.32	89.00 ± 1.25	50.28	30.64 ± 3.72
	256	50.56	89.40 ± 2.50	50.52	99.45 ± 0.33	50.54	89.60 ± 1.23	50.46	34.20 ± 3.66
	512	50.94	89.60 ± 2.60	50.87	99.40 ± 0.33	50.90	90.25 ± 1.17	50.75	33.88 ± 3.78
	1024	51.57	90.00 ± 2.50	51.43	99.30 ± 0.35	51.51	90.25 ± 1.18	51.21	31.80 ± 3.60
	2048	52.57	89.60 ± 2.60	52.34	99.45 ± 0.30	52.47	90.25 ± 1.13	51.95	32.04 ± 3.60
	4096	54.08	89.40 ± 2.70	53.72	99.35 ± 0.35	53.94	90.25 ± 1.18	53.06	33.04 ± 3.82
SANTA++ (K-means)	128	15.91	16.80 ± 3.20	16.63	94.20 ± 1.10	16.83	81.60 ± 1.63	15.28	24.00 ± 3.44
	256	19.00	45.00 ± 4.50	20.34	97.65 ± 0.67	20.58	85.05 ± 1.50	18.10	27.52 ± 3.64
	512	24.16	61.80 ± 4.20	26.33	98.25 ± 0.60	26.58	86.65 ± 1.25	22.85	28.60 ± 3.44
	1024	32.24	75.00 ± 3.80	35.34	98.65 ± 0.47	35.65	87.80 ± 1.25	30.37	28.36 ± 3.68
	2048	44.25	86.60 ± 2.90	48.11	99.30 ± 0.35	48.44	89.15 ± 1.25	41.78	26.08 ± 3.54
	4096	61.04	89.00 ± 2.60	64.75	99.20 ± 0.38	65.06	89.80 ± 1.20	58.62	31.84 ± 3.56
SANTA++ (Contiguous)	128	15.05	1.80 ± 1.10	15.07	89.20 ± 1.27	15.14	70.25 ± 2.03	14.38	31.76 ± 3.42
	256	17.27	12.40 ± 2.90	17.37	93.15 ± 1.15	17.45	77.00 ± 1.58	16.24	42.12 ± 3.88
	512	21.19	35.80 ± 4.20	21.46	95.65 ± 1.08	21.54	80.75 ± 1.33	19.75	43.68 ± 3.90
	1024	27.73	60.20 ± 4.20	28.33	97.70 ± 0.68	28.42	85.30 ± 1.25	25.88	39.96 ± 3.72
	2048	38.13	73.20 ± 3.70	39.19	98.50 ± 0.52	39.25	87.75 ± 1.25	35.72	35.60 ± 3.82
	4096	53.71	85.20 ± 3.10	55.12	99.15 ± 0.38	55.22	88.85 ± 1.20	50.44	28.40 ± 3.68

Method	S	CWE KV (%)	CWE Acc (%)	FWE KV (%)	FWE Acc (%)	QA-1 KV (%)	QA-1 Acc (%)	QA-2 KV (%)	QA-2 Acc (%)
SDPA	—	100.00	74.34 ± 1.92	100.00	91.87 ± 1.30	100.00	69.80 ± 3.90	100.00	46.00 ± 4.50
SANTA	128	50.31	70.20 ± 1.91	50.41	89.73 ± 1.43	50.36	69.40 ± 4.00	50.34	48.20 ± 4.50
	256	50.53	71.86 ± 1.94	50.73	91.67 ± 1.30	50.62	69.20 ± 4.00	50.59	47.00 ± 4.40
	512	50.89	73.72 ± 1.89	51.27	91.47 ± 1.37	51.08	70.40 ± 4.00	51.01	46.80 ± 4.60
	1024	51.50	73.98 ± 1.98	52.17	91.67 ± 1.33	51.84	70.00 ± 3.90	51.71	46.80 ± 4.30
	2048	52.45	75.02 ± 1.79	53.62	91.40 ± 1.40	53.02	70.00 ± 3.80	52.82	46.80 ± 4.40
	4096	53.88	74.18 ± 1.85	55.80	91.53 ± 1.33	54.81	69.60 ± 3.90	54.49	46.20 ± 4.40
SANTA++ (K-means)	128	16.13	56.80 ± 1.80	16.02	77.67 ± 2.13	17.06	65.00 ± 4.00	16.90	46.20 ± 4.20
	256	19.71	59.08 ± 1.98	19.48	78.80 ± 2.13	21.20	69.00 ± 4.10	20.95	47.00 ± 4.50
	512	25.44	62.94 ± 1.90	25.22	82.33 ± 1.67	27.64	69.60 ± 4.10	27.24	46.80 ± 4.30
	1024	34.14	67.26 ± 2.02	34.25	85.07 ± 1.50	37.09	68.60 ± 4.00	36.60	47.20 ± 4.40
	2048	46.58	70.34 ± 1.97	47.47	89.33 ± 1.40	50.10	70.20 ± 4.00	49.62	47.80 ± 4.30
	4096	63.25	72.76 ± 1.84	65.20	90.07 ± 1.33	66.79	70.20 ± 3.90	66.42	45.60 ± 4.50
SANTA++ (Contiguous)	128	14.98	41.32 ± 1.38	14.92	83.87 ± 1.73	15.28	63.80 ± 4.20	15.25	44.20 ± 4.50
	256	17.23	47.18 ± 1.43	17.28	84.93 ± 1.60	17.82	65.00 ± 4.10	17.83	45.20 ± 4.20
	512	21.26	51.86 ± 1.76	21.52	85.47 ± 1.83	22.24	66.80 ± 3.90	22.29	47.00 ± 4.60
	1024	28.01	58.00 ± 1.70	28.75	88.20 ± 1.63	29.48	69.00 ± 3.70	29.65	48.60 ± 4.40
	2048	38.65	63.42 ± 1.98	40.37	91.53 ± 1.33	40.70	70.20 ± 3.90	41.05	48.60 ± 4.30
	4096	54.22	69.96 ± 2.03	57.46	91.13 ± 1.43	56.90	70.20 ± 3.90	57.42	48.20 ± 4.70

Table 12: **Per-task accuracy and KV memory access on RULER (32K context) on Qwen2.5-7B-Instruct.** *KV* is the theoretical GQA-aware KV memory access (%), and S denotes the nominal budget. Accuracy values are reported as mean ± 95% bootstrap confidence interval half-width.

D TRITON DECODE KERNELS

The reference team construction in Section 2 records member indices and actual-key representatives without reordering the original KV cache. For this GPU benchmark, preparation creates separate packed copies of the prompt keys and values so each team's members occupy contiguous ranges; the original cache remains unchanged. For the captured 32K FP16 workload, packed K/V add 64.0 MiB per layer, frozen representative keys add 7.73 MiB, and team starts/lengths add 0.24 MiB, for 71.98 MiB of persistent preparation storage beyond the original K/V, excluding kernel workspaces. Construction and packing occur before the five timed launches below. Here K denotes the number of teams selected per query head.

1. Grouped representative scorer. The first kernel computes each query's dot product with one representative key per team. Query heads sharing a KV head are processed together so that representative-key loads can be reused, but the kernel writes separate scores for each query head. It reads neither values nor the other member keys at this stage.

2. Parallel per-query top-K. To parallelize candidate selection, the second kernel processes blocks of teams independently. It scales the representative scores by $1/\sqrt{d}$ and adds the logarithm of team size to form the routing logits. Independent Gumbel noise for each query-head–team pair gives Gumbel-perturbed routing scores. Each block retains its largest $K + 1$ scores and their team indices. For the timed configuration, $K = 31$, so this gives a power-of-two working set of 32 scores. The extra $(K + 1)$st score supplies the conditional-inclusion threshold for the K sampled teams.

3. Fused exact merge and GQA union. For each query head, the third kernel merges the local candidate lists to recover the global top-K teams and the $(K + 1)$st score. It uses this extra score and the unperturbed routing logits to compute the selected teams' conditional inclusion corrections (Equation 7). In the same launch, it forms the union of teams selected by query heads sharing a KV head. Each team appears once, with metadata identifying its member range and the heads that selected it. This shared read plan preserves each head's selection and corrections.

4. Grouped selected attention. The fourth kernel reads the union's member keys and values and computes their exact query–key scores. Reads are shared across query heads, but each head accumulates contributions only from its own selected teams. Each team's inverse-inclusion weight is applied to both the unnormalized weighted value sum and the attention mass. The kernel does this by subtracting $\log c_g$ from each selected member's logit, so the same correction enters both sums. To evaluate selected rows in parallel, the kernel divides them into disjoint ranges. Each range produces partial sums and the softmax scaling information needed to combine them.

5. Split-softmax reducer. Partial results may use different softmax scales, so the final kernel first rescales them to a common scale for each query head. It then adds the weighted value sums and attention masses and divides the former by the latter to produce the attention output. This kernel reads only partial results.

E KERNEL EVALUATION PROTOCOL AND PROFILING

Captured workload. The layer-14 Qwen2.5-7B-Instruct input used in Table 5 has 32,768 cached tokens, batch size one, one query token, 28 query heads, four KV heads, head dimension 128, and FP16 inputs. Seven query heads share each KV head. The operator selects $K = 31$ teams per query head (nominal budget $S = 124$). The number of selected rows varies with team size. The query, prompt cache, and team structure remain fixed, with no appended suffix. The captured team partition was constructed with k-means parents. Prefill, team construction, cache packing, input preparation, and workspace allocation are excluded from timing.

Device, software, and baseline. Measurements use an NVIDIA GeForce RTX 5090 Laptop GPU, NVIDIA driver 610.74, CUDA runtime 13.0, PyTorch 2.12.1+cu130, Triton 3.7.1, and Python 3.12.14. TF32 matrix multiplication is disabled. The dense baseline is PyTorch SDPA with the

Flash backend explicitly forced and native GQA enabled. Its observed execution consists of split-KV forward and combine launches. SANTA++ timing covers all five launches in Appendix D.

Sampled-reference validation. The reported pre-timing checks cover random-number generation, local top-K selection and exact merging, inclusion corrections, GQA union construction, and buffer guards. Outputs are compared with an independent reference using identical selected rows and conditional inclusion corrections. The reported maximum absolute output error across three checked sampling realizations is less than 4.30×10^{-6}. The dense Flash result is checked separately, and the observed kernel catalog verifies the five-launch sparse and two-launch dense paths.

Profiler-off timing. Canonical latency uses CUDA events with profiling disabled and without CUDA Graph capture. We collect 20 paired rounds, each preceded by 300 warm-up pairs and containing 1,000 complete-operator observations per method, for 20,000 observations per method overall. Measurements are interleaved, with the starting method reversed between adjacent rounds. Before each measured invocation, a 512 MiB eviction buffer is read and written outside the timed interval to reduce repeated-input cache reuse. There is no eviction between kernels within an invocation, so intermediate data can be reused. The event interval includes GPU execution and device idle gaps between host-issued launches, rather than host wall-clock latency. Implementation and launch parameters remain fixed throughout collection.

Aggregation and variability. For each method, we take the median of its 1,000 observations within each round, then the median of the 20 round medians. Speedup is the ratio of the Flash and SANTA++ estimates. The reported paired-round bootstrap uses 10,000 resamples and gives a 95% interval of $[1.689, 1.691]$ for this ratio, preserving the two methods' within-round correspondence when resampling rounds. This interval summarizes within-session uncertainty in the speedup ratio. Round medians span 63.42–100.90 μs for SANTA++ and 105.38–168.14 μs for Flash. Both methods are slower in the first two rounds. All 20 rounds are retained.

Isolated-stage measurements. Each stage has 500 observations per round, or 10,000 overall. Upstream eviction precedes the timed stage, and stage order rotates across measurements. Table 13 reports medians of round medians. These stages have different cache and timing boundaries from the uninterrupted operator, so their medians must not be summed or interpreted as additive percentages of its latency. The final row is a separate direct measurement of all five launches.

Table 13: Profiler-off SANTA++ timings for independently measured regions of the captured 32K workload. Entries are medians of round medians; the complete operator is measured directly.

Directly timed region	Median latency (μs)
Grouped representative scorer	20.32
Parallel per-query top-K	15.51
Fused exact merge + GQA union	16.16
Grouped selected attention	19.97
Split-softmax reducer	9.95
Complete five-launch SANTA++	**63.52**

NCU replay and metric definitions. Nsight Compute (NCU) 2025.3.1 provides separate diagnostic measurements. Each row in Table 14 reports per-metric medians from three captures of the same workload and a fixed sampling realization, using kernel replay, cache control `all`, and clock control `none`. Replay durations do not replace canonical profiler-off latency. CTAs are launched thread blocks; *Occ.* is achieved active-warps occupancy; *Elig.* is the average number of eligible warps per scheduler per active cycle; and *Reg./thr.* is register allocation per thread. *DRAM (%)* reports throughput as a percentage of peak sustained throughput.

The profiler distinguishes resource use across stages. Dense split-KV forward reports 83.95% DRAM throughput, compared with 63.85% for representative scoring and 19.46% for selected attention. Top-K achieves 84.51% occupancy, whereas merge/union launches only four blocks.

Table 14: Nsight Compute measurements for the captured 32K, batch-one workload. Entries are per-metric medians of three independent captures of a fixed sampling realization; durations use kernel replay.

Kernel	CTAs	NCU dur. (μs)	Occ. (%)	Elig.	DRAM (%)	Reg./thr.
SANTA++ representative scorer	248	14.14	24.49	0.08	63.85	56
SANTA++ top-K	868	10.78	84.51	2.40	10.61	30
SANTA++ merge + union	4	10.94	16.62	0.38	1.59	44
SANTA++ selected attention	128	14.30	12.88	0.20	19.46	144
SANTA++ reducer	28	4.80	8.33	0.05	11.07	40
Flash split-KV forward	128	89.92	8.33	0.10	83.95	200
Flash split-KV combine	7	21.92	8.30	0.03	3.04	44

F TOKEN SPARSITY AFTER LATENT COMPRESSION

Multi-head latent attention (MLA) in DeepSeek-V2-Lite-Chat already compresses the representation stored for each token and shares it across all 16 attention heads. Each cache row contains one 512-dimensional normalized content latent and one 64-dimensional rotary-position key. We test whether useful token sparsity remains after this compression and sharing. SANTA++ selects which token rows to evaluate for a query rather than further compressing each row. With a shared cache, a small per-head selection need not give a small shared-row union: a row is omitted from selected attention only when no head selects it.

We evaluate four budgets, $S \in \{128, 256, 512, 1024\}$, on three RULER tasks at 8K context, using 50 matched prompts per task. The dense reference uses SDPA through the same compressed-cache adapter and precision settings as SANTA++. Access is counted in shared latent rows, including both routing and the union of selected rows across heads, as defined below.

Accuracy–access trade-off. At $S = 512$, SANTA++ reaches a mean accuracy of 84.56% versus 86.22% for SDPA, retaining 98.07% of the dense mean with 86.18% logical shared-row access (Table 15). This reduces logical access by a further 13.82% relative to an already shared latent cache. A lower budget trades more accuracy for fewer reads: $S = 256$ retains 91.56% of the dense mean at 70.09% logical access. These results suggest that selective token access can complement latent compression and head sharing.

Method	S	KV (%)	Acc (norm) (%)	Acc (abs) (%)
SDPA	—	100.00	100.00	86.22 ± 4.56
SANTA++ (Latent k-means)	128	56.28	80.80	69.67 ± 5.92
	256	70.09	91.56	78.94 ± 5.14
	512	86.18	98.07	84.56 ± 4.72
	1024	101.89	97.81	84.33 ± 5.00

Table 15: **Token sparsity in a shared-latent cache at 8K.** DeepSeek-V2-Lite-Chat, with 50 matched prompts per task on `niah_multivalue`, `fwe`, and `qa_1`. Absolute accuracy is the equal-weight task mean; normalized accuracy divides it by the SDPA mean, times 100. KV denotes logical shared-row access, including routing, under Equation 14. Throughout these tables, $\pm$ gives half the span of a 95% percentile-bootstrap interval using 10,000 resamples. Prompts are resampled within each task, retaining task strata for the aggregate. Interval endpoints need not be symmetric about the point estimate.

Shared-row accounting. For a layer-level decoding call c, let $\mathcal{I}_{c,h}$ contain head h's selected prompt members and exact generated suffix. Let r_c charge one routing row per representative, and let d_c be the current dense shared-row count. We report

$$\mathrm{KV}_{\mathrm{MLA}}\ (\%) = 100\,\frac{\sum_c \left(r_c + \left|\bigcup_h \mathcal{I}_{c,h}\right|\right)}{\sum_c d_c}. \tag{14}$$

One counted row represents a token's shared content latent and rotary-position key, without separate charges for an expanded key and value. The union counts an attention row once across heads without changing any head's estimator. A row used in both routing and attention is counted in both passes. Counts are pooled across prompts, layers, and decoding steps, and across tasks for the aggregate, before division.

Each method's denominator is the dense shared-row count along its own generation trajectory, not the total from the separate SDPA run. Generation length and the sparse first-token replay described below can therefore change the denominator. The reference implementation stores expanded routing keys and reconstructs full K/V before selection. These measurements describe logical access opportunities, not realized memory-traffic savings or a latent-cache speedup.

Head sharing and routing limit the reduction. Independent head selections can cover a large fraction of the shared cache even when each head selects few teams. At $S = 512$, the attention union and exact suffix account for 63.75% of the dense shared-row count, and routing adds 22.43%. At $S = 1024$, the attention term grows to 79.51%; including routing raises total logical access to 101.89%, without improving the aggregate accuracy point estimate. A value above 100% is possible because routing and attention are separate passes. Thus, increasing the per-head budget can consume the remaining shared-row savings before it improves task accuracy.

Task dependence. The additional sparsity is useful on some tasks at a smaller accuracy cost than on others (Tables 16–18). At $S = 512$, `fwe` matches SDPA's 96.67% score with 80.34% logical access. Multivalue retrieval is one percentage point below its dense score, whereas `qa_1` remains four points below. Increasing the budget to $S = 1024$ raises the `qa_1` point estimate but lowers the multivalue-retrieval estimate. The full sweep therefore shows a task-dependent trade-off rather than a monotonic accuracy gain from reading more rows.

Method	S	KV (%)	Acc (norm) (%)	Acc (abs) (%)
SDPA	—	100.00	100.00	96.67 ± 2.67
SANTA++ (Latent k-means)	128	47.57	84.82	82.00 ± 7.00
	256	62.25	94.48	91.33 ± 5.33
	512	80.34	100.00	96.67 ± 3.00
	1024	98.45	99.31	96.00 ± 3.00

Table 16: **RULER `fwe` at 8K.** Accuracy on 50 matched prompts per method. Normalized accuracy divides the reported two-decimal task score by SDPA's score on this task, times 100. Access and uncertainty follow Table 15.

Method	S	KV (%)	Acc (norm) (%)	Acc (abs) (%)
SDPA	—	100.00	100.00	94.00 ± 4.00
SANTA++ (Latent k-means)	128	57.88	77.66	73.00 ± 8.75
	256	71.89	93.09	87.50 ± 5.50
	512	87.67	98.94	93.00 ± 4.75
	1024	102.64	92.55	87.00 ± 7.75

Table 17: **RULER `niah_multivalue` at 8K.** Accuracy on 50 matched prompts per method. Normalized accuracy divides the reported two-decimal task score by SDPA's score on this task, times 100. Access and uncertainty follow Table 15.

Model, precision, and generation. Weights use bitsandbytes LLM.int8 quantization, except for the latent-to-K/V projection and language-model output head. These unquantized modules and the cache use BF16. Routing scores and sampled attention reductions use FP32. The dense reference uses SDPA through the same compressed-cache adapter, rather than the unmodified checkpoint's generation path. SDPA may select its Flash or memory-efficient backend; the math backend is disabled.

Method	S	KV (%)	Acc (norm) (%)	Acc (abs) (%)
SDPA	—	100.00	100.00	68.00 ± 13.00
SANTA++ (Latent k-means)	128	58.41	79.41	54.00 ± 14.00
	256	72.81	85.29	58.00 ± 14.00
	512	88.50	94.12	64.00 ± 13.00
	1024	103.87	102.94	70.00 ± 13.00

Table 18: **RULER `qa_1` at 8K.** Accuracy on 50 matched prompts per method. Normalized accuracy divides the reported two-decimal task score by SDPA's score on this task, times 100. Access and uncertainty follow Table 15.

All methods use the same prompts, answer references, and chat formatting. Decoding is greedy and stops at the end-of-sequence token or at 50, 128, and 32 new tokens for `fwe`, `niah_multivalue`, and `qa_1`, respectively. Data generation and prompt selection use seed 42, and k-means uses seed 0. Sampling uses deterministic prompt-specific seeds derived from base seed 0 and each prompt's identifier, shared between budgets. Prompt scores count the fraction of required answers recovered for `fwe` and `niah_multivalue`; `qa_1` checks whether any accepted answer appears in the output.

Teams and head-specific selection. Minibatch k-means and representative assignment use the content latents after the checkpoint's normalization, without additional unit-norm normalization or the rotary components. This produces one partition per layer shared by all heads. Representatives are actual members selected within each parent, and each representative is assigned to its own team.

Each head scores a representative using its query and that head's reconstructed representative key, including the rotary-position component, with the checkpoint's attention scale. These routing keys are frozen after dense prefill. Selected contributions use current-cache keys and values. Heads share teams but draw independently and retain their own conditional inclusion corrections (Section 2). We use $P = 16$ and $R = 4$, so the four nominal budgets request 32, 64, 128, and 256 teams per head, respectively, capped by availability.

Exact generated suffix. The fixed teams cover the entire prompt. After dense prefill, the sparse adapter removes the last prompt entry from the cache and replays that token at its original position through sparse attention. The first generated token therefore depends on sparse attention as well. There is no exact prompt tail: the exact suffix is empty on the first sparse call and grows with newly generated tokens, which are not added to the fixed teams.