Dropping the Anchor: Statistical Context Summarization for Distributed Systems via Pulsar Attention

Aryan Sood [1] **Shantanu Acharya** [2]

Abstract

Inference with large language models (LLMs) on long sequences is computationally expensive due to the quadratic complexity of self-attention. Distributed blockwise methods such as Star Attention reduce this cost by sharding context across hosts, but rely on prepending a static, content-blind copy of the first block to every host. We propose Pulsar Attention, which replaces the static anchor with two lightweight, content-aware components: a small attention-sink prefix that stabilizes softmax, and compact cross-block summaries built via a Max-IDF heuristic that selects chunks containing globally rare tokens. This reduces the Phase 1 per-GPU FLOPs by up to 3.3× over Star Attention while retaining an identical KV cache footprint. On RULER and BABILong with Llama-3.1-8B, Pulsar Attention outperforms both Star Attention and dense attention at sequence lengths up to 128K tokens, with absolute gains of up to 4.7% over the dense baseline.

1. Introduction

Large Language Models (LLMs) have demonstrated strong performance on tasks requiring long context windows, but scaling inference remains expensive: self-attention is quadratic in sequence length, and the KV cache grows linearly, quickly exhausting device memory on a single GPU.

Distributed methods address this by sharding context across GPUs. Ring Attention (Liu et al., 2023) computes exact global attention by circulating KV blocks in a ring, but requires coordinated communication at every layer. Star Attention (Acharya et al., 2025) eliminates this communication during context encoding by processing independent blocks in parallel, recovering global attention at query time through a lightweight softmax merge. To prevent softmax collapse, each host prepends a static copy of the first block (called the anchor block) to its input.

The anchor is effective but it duplicates the same tokens regardless of relevance, doubling per-host sequence length and FLOPs during context encoding. Because it is fixed to the sequence start, a block i receives no information about the intermediate blocks, which widens as sequences grow.

We propose Pulsar Attention, which replaces the static anchor with two lightweight, content-aware components: (i) a small attention-sink prefix (64 tokens) that stabilizes softmax at a fraction of the anchor's cost, and (ii) compact Max-IDF block summaries that select contiguous chunks containing globally rare tokens from each preceding block. Summaries propagate causally and adapt to each block's content, prioritizing tokens most likely to carry task-critical information. Our main contributions are:

- We replace the static anchor with content-aware summaries and attention sinks, reducing Phase 1 per-GPU FLOPs by up to 3.3× over Star Attention while retaining an identical KV cache footprint.
- We introduce Max-IDF chunk scoring and validate it against three alternative heuristics on BABILong, showing rare-token selection best captures cross-block context effectively.
- We show on RULER that Pulsar Attention outperforms both Star Attention and dense attention on context lengths upto 128K, with gains of up to +4.7% over the dense baseline.

2. Background

Transformer-based LLMs cache key and value vectors for all previous tokens during autoregressive decoding. While this avoids redundant computation, KV cache memory grows linearly with sequence length, quickly exhausting the device memory at long contexts.

Liu et al. (2023) addresses this by distributing the KV cache across GPUs and circulating blocks in a ring to compute ex-

[1]Department of Electrical Engineering, Indian Institute of Technology Roorkee, India [2]NVIDIA. Correspondence to: Aryan Sood <aryan_s2@ee.iitr.ac.in>, Shantanu Acharya <shantanua@nvidia.com>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

act global attention. This enables arbitrarily long sequences but requires cross-host communication at every layer.

Acharya et al. (2025) eliminates this communication during context encoding with a two-phase design. In Phase 1, the context is split into contiguous blocks processed independently on parallel hosts; a static copy of the first block (the anchor block) is prepended to each host's input to prevent softmax collapse. In Phase 2, the query is broadcast to all hosts, which compute local attention scores and aggregate them via an online softmax merge to recover global attention. While effective, the anchor duplicates the same tokens regardless of relevance, doubling per-host sequence length and FLOPs during Phase 1. Our method addresses this limitation.

3. Methodology

3.1. Phase 1: Statistical Context Encoding

Like Star Attention, Pulsar Attention partitions the input sequence of length L into n contiguous blocks $\{B_0, B_1, \ldots, B_{n-1}\}$, each assigned to a parallel host (Figure 1a). Rather than duplicating the entire first block as a static anchor, we replace it with two lightweight, content-aware components: an attention-sink prefix and Max-IDF block summaries. Context encoding proceeds in four steps.

Attention sinks. The first s_{sink} tokens of B_0 are extracted as a fixed prefix $\mathcal{S}_{\text{sink}}$ (default $s_{\text{sink}} = 64$). Following Xiao et al. (2023), these tokens stabilize the softmax distribution during blockwise attention at a fraction of the cost of a full anchor block.

Summary generation. Before any neural computation, every host constructs an identical corpus-level IDF table from the token IDs of the full sequence, an $O(L)$ integer-only operation that requires no inter-host communication and no GPU time. Using this table, each host scores non-overlapping contiguous chunks of m tokens (default $m = 32$) within each block via a Max-IDF heuristic:

$$\text{score}(C) = \max_{t \in C} \text{IDF}(t) \tag{1}$$

$$\text{IDF}(t) = \log\left(\frac{n}{\max(\text{df}(t), 1)}\right) \tag{2}$$

where $\text{df}(t)$ is the number of blocks containing token t. The top-k chunks per block are selected and concatenated in positional order to form block summary Σ_i, with a per-block budget of σ tokens (default $\sigma = 0.125 \times |B_i|$). We choose contiguous chunks rather than individual tokens to preserve local syntactic structure in the resulting key–value representations. Max-IDF is preferred over averaging heuristics (TF-IDF, BM25) because retrieval-critical information like UUIDs, entity names, numerical values are typically concentrated in a single rare token per chunk; averaging dilutes this signal by a factor of $1/m$ (see Section 5.1 for an empirical comparison).

Algorithm 1 Pulsar Attention

Input: Blocks $\{B_0, \ldots, B_{n-1}\}$, query Q, s, m, k, T
Output: Generated token sequence
$\mathcal{S}_{\text{sink}} \leftarrow B_0[0{:}s]$; build $\text{IDF}(t)$ from $\text{df}(t)$ across all blocks
for host $i = 0, \ldots, n{-}1$ **in parallel do**
  $\Sigma_i \leftarrow$ top-k chunks of B_i by $\max_{t \in C} \text{IDF}(t)$, in positional order
  $\text{Input}_i \leftarrow [B_0]$ if $i{=}0$, else $[\mathcal{S}_{\text{sink}} \,\|\, \Sigma_0 \,\|\, \cdots \,\|\, \Sigma_{i-1} \,\|\, B_i]$
  $\text{KV}_i \leftarrow \textsc{ForwardPass}(\text{Input}_i)$; discard non-$B_i$ KV entries
end for
for $t = 1$ **to** T **do**
  $(A_h, \ell_h) \leftarrow \textsc{FlashAttn}(q, \text{KV}_h)$ for each host h in parallel; `all_gather` $\{(A_h, \ell_h)\}$
  $(A, \ell) \leftarrow \textsc{OnlineSoftmaxMerge}\big(\{(A_h, \ell_h)\}_{h=0}^{n-1}\big)$
  $q \leftarrow \text{argmax}(\textsc{LMHead}(A))$; append $\text{KV}(q)$ to KV_{n-1}; **if** $q =$ `EOS` **then break**
end for
return generated sequence

Causal Assembly. Each host i prepends only summaries from causally preceding blocks along with $\mathcal{S}_{\text{sink}}$, forming the augmented Phase 1 input:

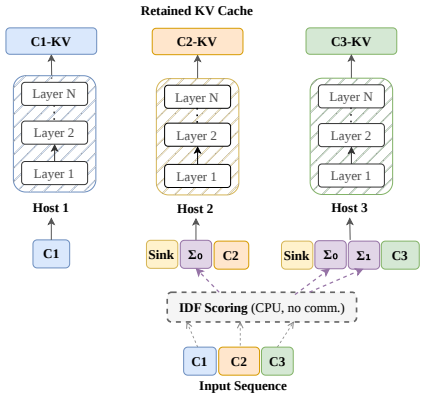
$$\text{Input}_i = \begin{cases} [B_0] & i = 0 \\ [\mathcal{S}_{\text{sink}} \| \Sigma_0 \| \cdots \| \Sigma_{i-1} \| B_i] & i > 0 \end{cases} \tag{3}$$

Each token retains its original global position ID, preserving the inter-token distances encoded by RoPE (see Appendix E for a comparison with contiguous re-numbering).

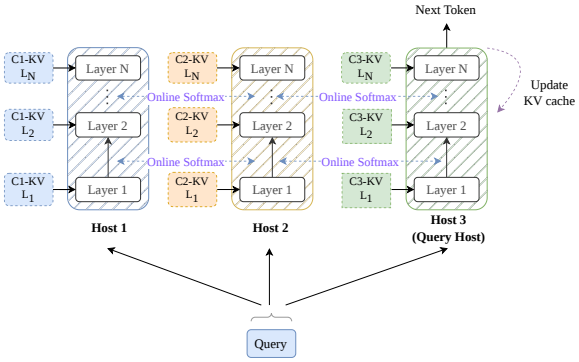
KV Discard. After the forward pass, KV states for $\mathcal{S}_{\text{sink}}$ and $\{\Sigma_j\}_{j<i}$ are discarded; only the $|B_i|$ entries belonging to the block itself are retained. The final distributed KV cache spans exactly L tokens with no duplication, and peak per-host memory during Phase 1 is strictly lower than Star Attention's: each host processes at most $L/n + s_{\text{sink}} + C$ tokens versus Star's $2L/n$, where $C \ll L/n$ is the total summary budget.

3.2. Phase 2: Global Query Encoding and Generation

Because Phase 1 produces a clean, non-overlapping distributed KV cache, Phase 2 is structurally identical to Star Attention (Acharya et al., 2025): the query is broadcast to all hosts, each computes local attention via FlashAttention, and a designated query host aggregates the results through an online softmax merge (Milakov & Gimelshein, 2018). The full pseudocode is given in Algorithm 1.



(a) Phase 1: Statistical context encoding. Host i assembles input $[\mathcal{S}_{\text{sink}} \parallel \Sigma_0 \parallel \cdots \parallel \Sigma_{i-1} \parallel B_i]$, runs a forward pass, then discards sink and summary KV entries, retaining only KV(B_i).



(b) Phase 2: Global query encoding. Each host attends over its retained KV shard; an `all_gather` and online softmax merge (Milakov & Gimelshein, 2018) recovers exact global attention without transmitting the full KV cache.

Figure 1. Pulsar Attention overview. *(Left)* Phase 1 replaces the static anchor block with a content-aware prefix: a small attention-sink and Max-IDF statistical summaries of all causally preceding blocks. *(Right)* Phase 2 broadcasts the query to all hosts and merges local attention scores into an exact global output via online softmax.

4. Experiments

4.1. Experimental Setup

Model. We evaluate on Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024), which supports a native 128K context window. All experiments use bfloat16 precision with FlashAttention-2. Generation uses greedy decoding.

Distributed configuration. Following Star Attention, the context is partitioned into $B = 4$ equal blocks, with one block assigned per GPU. All distributed experiments use 4 hosts for all context lengths. However we switched to 8 blocks for 128K due to memory efficiency. The summary budget is fixed at 12.5% of block size (chunk size m=32 tokens, with the number of selected chunks k scaled proportionally to context length). The attention-sink prefix is fixed at 64 tokens across all settings.

Benchmarks. We evaluate on two benchmarks: (i) RULER (Hsieh et al., 2024), a synthetic long-context benchmark comprising 13 tasks across five categories (single-needle retrieval, multi-needle retrieval, multi-hop tracing, aggregation, and question answering), with 500 samples per task per context length; and (ii) BABILong (Kuratov et al., 2024), a reasoning benchmark requiring multi-step inference over supporting facts embedded in long distractor text, with 1,000 samples per task. We report exact-match accuracy for both benchmarks.

Baselines. We compare against four baselines: (i) Full (Dense) Attention, computed on a single GPU as the accuracy ceiling; (ii) Star Attention (Acharya et al., 2025), the most directly comparable distributed method; (iii) StreamingLLM (Xiao et al., 2023), a sparse method combining sink tokens with sliding-window attention; and (iv) MInference (Jiang et al., 2024a), which dynamically selects per-head sparse attention patterns. Baselines marked with † are reproduced from Acharya et al. (2025) using the same model and hardware configuration.

4.2. Results on RULER

Table 1 reports average accuracy across all 13 RULER tasks at context lengths from 16K to 128K. Pulsar Attention outperforms Star Attention at 32K (+2.8%), 64K (+6.1%), and 128K (+4.7%), with the gap widening at longer contexts. At 16K, Pulsar trails Star Attention by 0.9%, likely because the summary budget (512 tokens from a 4K block) provides less coverage than Star's full 4K anchor at short contexts where the anchor's cost is relatively low.

Notably, Pulsar also exceeds Full Attention accuracy at 32K (+4.0%), 64K (+4.7%), and 128K (+2.8%). We hypothesize that block-local attention with content-aware summaries produces sharper softmax distributions than full-sequence attention, which increasingly dilutes signal across irrelevant tokens at long contexts.

Among the single-GPU baselines, MInference performs competitively at 16K-64K but degrades sharply to 58.2% at 128K (-21.3 points versus Pulsar), suggesting that its offline sparse pattern estimation becomes unreliable at extreme lengths. StreamingLLM collapses past 16K, confirming that recency-biased context windows cannot substitute for

Table 1. RULER average accuracy (%) on Llama-3.1-8B. †From Acharya et al. (2025), same model and setup. **Bold** = best among efficiency-focused methods.

Method	Setting	16K	32K	64K	128K	Avg
Full Attention	Single GPU	92.2	87.5	84.8	76.3	85.2
StreamingLLM†	Single GPU	74.8	48.6	26.2	30.8	45.1
MInference†	Single GPU	93.3	86.5	84.9	58.2	80.7
Star Attention†	Multi-GPU	91.3	88.7	83.4	74.4	84.4
Pulsar Attention (Ours)	Multi-GPU	90.4	**91.5**	**89.5**	**79.1**	**87.6**

Table 2. Per-task accuracy (%) at 128K on Llama-3.1-8B. NIAH MultiKey and NIAH Single scores are averaged across their three sub-tasks; QA averages SQuAD and HotpotQA.

Task	Dense	Star	Pulsar
NIAH Single (avg)	99.7	99.9	**100.0**
NIAH MultiKey (avg)	83.9	80.1	**78.7**
NIAH MultiValue	**91.6**	82.7	63.5
NIAH MultiQuery	**97.8**	96.6	97.0
Variable Tracking	61.8	**62.8**	61.0
Common Words	0.04	0.04	**81.2**
Frequent Words	72.3	75.9	**80.7**
QA (avg)	**58.8**	54.8	54.5
Overall Avg	76.3	74.4	**79.1**

cross-block information flow.

4.3. Results on BABILong

Table 3 reports accuracy on three BABILong tasks (qa1, qa3, qa5) at 16K context length using $B = 4$ blocks of 4K tokens each. Pulsar Attention with Max-IDF scoring ($s = 512$) achieves 37% on qa1, 44% on qa3, and 63% on qa5. On the multi-hop tasks (qa3 and qa5), Pulsar outperforms Star Attention (33% and 58% respectively) by +11 and +5 points. The advantage on qa3 is particularly notable: this task requires chaining three supporting facts that may reside in different blocks, and Max-IDF's ability to select chunks containing rare entity tokens provides cross-block context that Star's position-fixed anchor cannot. We present extended BABILong results across context lengths and scoring heuristics in the ablation study (Section 5.1).

4.4. RULER Task-Category Analysis

Table 2 reports per-task-category accuracy at 128K tokens on RULER, where the differences between methods are most pronounced.

Retrieval (NIAH). Single-needle retrieval is effectively saturated (100%) across all methods. On MultiKey and MultiQuery, Pulsar maintains near-parity with Dense, as Max-IDF reliably promotes chunks containing rare UUID or keyword needles. The notable exception is NIAH MultiValue (63.5% vs. Dense 91.6%): this task requires retrieving multiple values associated with a single key, where the values may reside in different blocks. Max-IDF selects the chunk containing the rare key identifier but may miss value chunks whose tokens are individually less distinctive.

Aggregation. The most striking result is Common Words Extraction, where both Full Attention (0.04%) and Star Attention (0.04%) effectively fail at 128K, while Pulsar achieves 81.2%. This task requires identifying tokens that appear frequently across the full sequence, precisely the global co-occurrence signal that diluted dense attention loses at long contexts and that a position-fixed anchor cannot reconstruct. Pulsar's causal summaries propagate recurring tokens from earlier blocks into each host's local context, restoring partial global frequency information without transmitting the full KV cache. Frequent Words Extraction shows the same pattern at a smaller magnitude (Pulsar 80.7% vs. Dense 72.3% vs. Star 75.9%).

Multi-hop reasoning. Variable Tracking and QA show broad parity across all three methods (within 4 points), consistent with the expectation that multi-hop inference benefits from cross-block context but ultimately requires denser inter-block communication than the current summary budget provides.

5. Ablations

5.1. Scoring Heuristic Comparison

Table 3 compares the four scoring heuristics against Even-Spaced selection and Star Attention's anchor on BABILONG-16K across three per-block summary budgets. Block size is fixed at 4 K tokens. (see Appendix A for task details).

(1) MAX-IDF achieves the best `qa5` score at $s = 128$ and is the only heuristic that *improves* with budget on the most discriminative task: `qa3` rises from 36 to 44 at $s = 512$, an 11-point margin over Star Attention.

(2) EVEN-SPACED records the highest `qa1` at $s = 128$ but degrades monotonically on `qa3` as budget grows ($36 \rightarrow 26 \rightarrow 25$), confirming that uniform coverage wastes tokens on low-information distractor spans once the budget is large

Table 3. BABILONG-16K accuracy (%) across three summary budgets ($B=4$, block size 4 K). **Bold**: best among the four statistical heuristics per group. Star Attention uses a fixed anchor block independent of summary budget.

	$s=128$			$s=512$			$s=1{,}024$		
Heuristic	qa1	qa3	qa5	qa1	qa3	qa5	qa1	qa3	qa5
TF-IDF	**40**	33	61	37	30	57	33	30	**60**
BM25	38	34	61	**39**	32	57	33	**35**	**60**
Entropy	36	**39**	65	38	39	62	35	27	**60**
Max-IDF *(ours)*	39	36	**66**	37	**44**	**63**	**36**	34	57
Even-Spaced	49	36	57	44	26	55	46	25	52
Star Attn	35	33	58	35	33	58	35	33	58

enough to exhibit selection bias.

(3) ENTROPY performs strongly at $s = 128$ (qa3: 39; qa5: 65) but collapses at $s = 1024$ (qa3: 27), indicating that diversity-based selection saturates when the budget approaches a large fraction of block size.

(4) All four statistical heuristics outperform Star Attention on qa5 at every budget, confirming that scored selection surpasses a static anchor.

Why Max-IDF excels at multi-hop tasks. Tasks qa3 and qa5 require joining multiple evidence spans, each mentioning a specific entity. These entity tokens receive the highest IDF values like proper nouns and task-specific names appear in only one or two blocks, yielding $\mathrm{IDF}(t) \approx \log N$. Max-IDF guarantees that any chunk containing a rare entity is selected regardless of surrounding distractor tokens. TF-IDF and BM25 *average* IDF over the chunk, so a single high-IDF entity among 31 filler tokens contributes only $\sim 3\%$ of the score. Entropy is blind to global rarity entirely.

Budget Selection. We used $s = 12.5\%$ of block size (512 tokens for a 4 K block), the configuration used in all RULER experiments. This budget provides sufficient coverage for multi-hop entity chains while remaining a small fraction of block size, limiting Phase 1 overhead.

5.2. Computational Efficiency

We derive Phase 1 memory and FLOPs analytically for Llama-3.1-8B (Grattafiori et al., 2024) ($L_{\text{layers}}=32$, $H_Q=32$, $H_{KV}=8$, $d_h=128$, fp16), using $B=4$, $s=512$, $k=64$. Under FlashAttention (Dao et al., 2022), HBM activation memory scales as $O(n)$ while FLOPs remain $\Theta(n^2 d)$.

Critical-path sequence length. The worst-case per-GPU input (last block with all prior summaries prepended) is:

$$n_{\text{Dense}} = L, \quad n_{\text{Star}} = \tfrac{L}{2}, \quad n_{\text{Pulsar}} = \tfrac{L}{4} + \underbrace{k + (B-1)\,s}_{=1{,}600}. \tag{4}$$

KV cache memory. Each token occupies 128 KB of KV cache. After Phase 1 KV discard, Star and Pulsar both retain only the local block, giving identical per-GPU KV cache:

$$M^{\text{KV}}_{\text{Star/Pulsar}} = \frac{L}{B} \times 128\,\text{KB}, \quad M^{\text{KV}}_{\text{Dense}} = L \times 128\,\text{KB}. \tag{5}$$

At $L=64$K: 2.15 GB per GPU (distributed) vs. 8.59 GB for Dense which is a $4\times$ per-GPU reduction exact and independent of s, k, or L.

Phase 1 FLOPs. FLOPs per layer for GQA attention are $\mathcal{F} = 2\,n^2\,(H_Q + H_{KV})\,d_h = 2\,n^2 \times 5{,}120$. Table 4 reports concrete values at three context lengths.

Star Attention's constant $4\times$ FLOPs reduction reflects its anchor always equalling one local block: $n^2_{\text{Star}}/n^2_{\text{Dense}} = 1/4$. Pulsar's growing advantage arises because the fixed prefix of 1,600 tokens represents a diminishing fraction of block size as L grows:

$$\frac{n^2_{\text{Star}}}{n^2_{\text{Pulsar}}} = \left(\frac{2L}{L+6{,}400}\right)^2 \xrightarrow{L\to\infty} 4\times. \tag{6}$$

At $L=64$K the ratio is already $3.3\times$.

Estimated wall-clock speedup. Star Attention (Acharya et al., 2025) reports wall-clock speedups of $1.1\times$, $1.2\times$, and $1.8\times$ over Dense at 16K, 32K, and 64K ($B=4$). Since Pulsar's Phase 2 is architecturally identical to Star's, its additional gain derives entirely from Phase 1:

$$r_{\text{Pulsar}} \;\approx\; r_{\text{Star}} \times \frac{n^2_{\text{Star}}}{n^2_{\text{Pulsar}}}. \tag{7}$$

The $4\times$ per-GPU KV cache reduction is exact across all L. The Phase 1 FLOPs advantage ($2.1\times$–$3.3\times$ over Star) compounds with Star's baseline to yield an estimated $6.0\times$ speedup over Dense at 64K. Summary scoring adds no overhead: IDF table construction requires $O(L)$ integer operations, a fraction below $10^{-6}\%$ of attention FLOPs.

Table 4. Phase 1 computational cost (Llama-3.1-8B, $B=4$, $s=512$, $k=64$, fp16). Activation memory and FLOPs are for the critical (last) block; KV cache is the per-GPU retained cache after Phase 1 discard.

	Critical-path length n			Peak act. mem. / layer			Phase 1 FLOPs / layer		
Method	16K	32K	64K	16K	32K	64K	16K	32K	64K
Dense (serial)	16,384	32,768	65,536	320 MB	640 MB	1.25 GB	2,749 G	10,995 G	43,981 G
Star Attn	8,192	16,384	32,768	160 MB	320 MB	640 MB	687 G	2,749 G	10,995 G
Pulsar *(ours)*	5,696	9,792	17,984	117 MB	201 MB	368 MB	332 G	982 G	3,312 G
Reduction vs. Dense									
Star Attn	2.0×	2.0×	2.0×	2.0×	2.0×	2.0×	4.0×	4.0×	4.0×
Pulsar *(ours)*	2.9×	3.4×	3.6×	2.9×	3.4×	3.6×	8.3×	11.2×	13.3×
Pulsar reduction vs. Star							2.1×	2.8×	3.3×

Table 5. Estimated Pulsar wall-clock speedup over Dense, derived from Eq. (7) with Star Attention's reported speedups (Acharya et al., 2025) as a calibration baseline.

	16K	32K	64K
r_{Star} over Dense	1.1×	1.2×	1.8×
$n^2_{\text{Star}}/n^2_{\text{Pulsar}}$	2.07×	2.80×	3.32×
Est. r_{Pulsar} over Dense	**∼2.3×**	**∼3.4×**	**∼6.0×**
KV cache	4.0×	4.0×	4.0×

6. Related Work

Distributed long-context inference. Sequence parallelism (Korthikanti et al., 2023) shards the sequence dimension with an all-reduce per layer. Ring Attention (Liu et al., 2023) removes this synchronization by overlapping communication with computation in a ring topology, but still requires cross-host transfers at every layer. Star Attention (Acharya et al., 2025) eliminates Phase 1 communication entirely by prepending a static first-block anchor to each host. Pulsar inherits Star's two-phase structure and communication-free Phase 1 but replaces the content-blind anchor with scored statistical summaries, recovering cross-block information without reintroducing communication.

Sparse and efficient attention. Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) combine sliding-window attention with global tokens at $O(n)$ cost but use fixed structural patterns. MInference (Jiang et al., 2024a) dynamically selects sparse attention patterns per head during pre-filling, achieving strong results up to 64K but degrading at 128K in our experiments (Table 1). FlexPrefill (Lai et al., 2025) extends dynamic sparsity with query-dependent budget allocation; its gains are orthogonal to our distributed encoding phase. StreamingLLM (Xiao et al., 2023) preserves attention-sink tokens for streaming generation; we adopt its sink mechanism as one component of our prefix but pair it with content-aware summaries rather than a recency-based sliding window. FlashAttention (Dao et al., 2022) provides the tiled block computation underlying all our Phase 1 forward passes.

KV cache compression and context selection. H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), and PyramidKV (Cai et al., 2024) evict low-importance KV entries during generation; these are orthogonal to Pulsar's Phase 1 and compatible to Phase 2. Prompt-compression methods such as LLMLingua (Jiang et al., 2023) and LongLLMLingua (Jiang et al., 2024b) use a proxy LM to score and remove tokens before the main forward pass. Pulsar's Max-IDF scoring is intentionally non-neural; it relies on corpus-level token statistics computable in a single CPU pass, adding negligible overhead while drawing on classical IDF-based retrieval (Robertson & Zaragoza, 2009).

7. Conclusion

We introduced Pulsar Attention, a drop-in replacement for Star Attention's static anchor block that uses two lightweight, content-aware components: a 64-token attention-sink prefix and Max-IDF block summaries at 12.5% of block size. On RULER with Llama-3.1-8B-Instruct, Pulsar outperforms both Dense and Star Attention at 32K–128K tokens, with gains of up to +4.7% over Dense and +6.1% over Star, while reducing Phase 1 per-GPU FLOPs by up to 3.3× relative to Star. The retained KV cache is identical to Star Attention's, making Pulsar compatible with existing Phase 2 infrastructure and post-hoc KV compression methods. The main limitation is on tasks requiring retrieval of multiple values per key (NIAH MultiValue), where single-score chunk ranking can miss value-bearing chunks that lack globally rare tokens. Future work will address this through query-aware or multi-score selection strategies, and validate the FLOPs-based speedup estimates with end-to-end wall-clock profiling.

References

Acharya, S., Jia, F., and Ginsburg, B. Star attention: Efficient LLM inference over long sequences. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Cai, Z., Zhang, Y., Gao, B., Liu, Y., Li, Y., Liu, T., Lu, K., Xiong, W., Dong, Y., Hu, J., et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, volume 35, pp. 16344–16359, 2022.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Jiang, H., Li, Y., Zhang, C., Wu, Q., He, X., Garg, S., Chen, Q., Jiang, J., Wei, Y., Chi, L., et al. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 2024a.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024b.

Korthikanti, V. A., Casper, J., Lym, S., McAfee, L., Andersch, M., Shoeybi, M., and Catanzaro, B. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.

Kuratov, Y., Bulatov, A., Anokhin, P., Rodkin, I., Sorokin, D., Sorokin, A., and Burtsev, M. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. *arXiv preprint arXiv:2406.10149*, 2024.

Lai, X., Lu, J., Luo, Y., Ma, Y., and Zhou, X. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

Milakov, M. and Gimelshein, N. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Robertson, S. and Zaragoza, H. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pp. 17283–17297, 2020.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

A. Dataset Details

A.1. BABILong

BABILong (Kuratov et al., 2024) extends the bAbI reasoning suite by embedding short synthetic reasoning chains inside long distractor text drawn from PG-19 books. The reasoning chains are generated from symbolic rules; the model must locate the relevant sentences among thousands of distractor tokens and apply multi-step logic. We evaluate on two tasks; Table 6 summarises their requirements.

Table 6. BABILong task descriptions.

Task	Description
`qa1`	**Single Supporting Fact.** One sentence states where a person currently is. The model must track the most recent location across multiple movement facts and answer "Where is X?"
`qa3`	**Three Supporting Facts / Object Path.** Three facts establish an object's path across locations. The model must answer "Where was the object *before* location Y?", requiring three-hop reasoning over the chain.
`qa5`	**Three-Argument Relations.** Facts involve object-transfer events (picked-up, gave, passed). The model answers queries such as "Who gave the apple?" or "What did X give to Y?", requiring entity-argument co-reference across three-place predicates.

Context splits cover $L \in \{16\text{K}, 32\text{K}, 64\text{K}, 128\text{K}\}$ tokens; each split contains 1,000 samples. We report results for `qa1`, `qa3`, and `qa5` as representative of increasing reasoning depth.

A.2. RULER

RULER (Hsieh et al., 2024) is a synthetic benchmark designed to test long-context recall under controlled conditions. We use all 13 tasks across four context lengths ($L \in \{16\text{K}, 32\text{K}, 64\text{K}, 128\text{K}\}$), 500 samples per task per length.

B. Chat Template and Prompt Format

All experiments use Meta-Llama-3.1-8B-Instruct with the standard Llama 3 instruct chat template. The full template skeleton is:

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>

You are a helpful assistant.<|eot_id|>
<|start_header_id|>user<|end_header_id|>

{task_instruction}

{in_context_examples}

{post_prompt}

<context>
{long_context}
</context>

Question: {query}<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Context/query split. Pulsar Attention (and Star Attention) separates the input into two distinct strings before tokenisation:

- **`prompt_context`**: everything from `<|begin_of_text|>` through the closing `</context>` tag, inclusive. This string is tokenised, split into blocks of size B, and processed by Phase 1.

Table 7. RULER task descriptions. Abbreviations: k = number of unique needle keys; v = values per key; r = number of queries.

Category	Task	Description
NIAH	Single-1	Single word key → single number value hidden in synthetic noise text.
	Single-2	Single word key → single number value hidden in Paul Graham essay excerpts.
	Single-3	Single word key → UUID value hidden in book text. UUID tokens are maximally rare (high IDF).
	MultiKey-1	4 distinct word keys all map to the same number; the model must retrieve the value from any key.
	MultiKey-2	An entire line of word keys maps to one value; model must match the full key phrase.
	MultiKey-3	UUID keys → UUID values; both key and value are rare tokens.
	MultiValue	1 key maps to 4 different numbers; the model must recall all of them.
	MultiQuery	4 queries each asking for the value of a different key from the same set of needles.
Tracking	Variable	A chain of variable assignments is hidden in noise (e.g., `x=3; y=x; z=y`). The model finds all variables assigned a target value.
Words	CWE	A numbered list of words is presented; some appear more often. The model identifies the 10 most common words.
	FWE	A "coded" text with high-frequency synthetic words. The model identifies the 3 most frequent codes.
QA	QA1	Reading comprehension over SQuAD (Rajpurkar et al., 2016) passages concatenated into a long context.
	QA2	Multi-hop reading comprehension over HotpotQA (Yang et al., 2018) documents; requires reasoning across two documents.

- **`prompt_query`**: the `Question: {query}` fragment followed by the assistant header tokens. This string is tokenised separately and processed by Phase 2.

Stop words. Generation halts at any of: `<|end_of_text|>`, `<|eom_id|>`, `<|eot_id|>`, or after `max_new_tokens` (128 for BABILong; task-dependent for RULER, see Table 9).

Tokenisation. Tokens are produced with `add_special_tokens=False`; the `<|begin_of_text|>` BOS token is included as part of the template string itself and must not be added a second time.

C. Scoring Heuristic Details

All heuristics operate on non-overlapping contiguous chunks of `chunk_size` tokens carved from each block. Chunks preserve local word order; the top-`num_chunks` chunks by score are selected and returned *in positional order* to maintain natural reading order for the Transformer.

C.1. Corpus-Level IDF Table

Before any scoring, a single pass over all blocks constructs a document-frequency table. Let N be the total number of blocks and $\mathrm{df}(t)$ the number of blocks containing token t:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{\max(\mathrm{df}(t),\, 1)}\right). \tag{8}$$

This is stored as a dense integer-indexed tensor `idf_table` of shape `(vocab_size,)`. All four IDF-based heuristics perform a single gather from this table; no Python loops over the vocabulary are required.

C.2. TF-IDF

$$\text{score}_{\text{TF-IDF}}(C) = \frac{1}{|C|} \sum_{p \in C} \text{IDF}(t_p). \tag{9}$$

This equals the classical mean TF-IDF over types, and permits a single vectorized lookup: `idf_table[chunk_tokens].sum() / len(chunk)`.

Strengths. Balanced, general-purpose; promotes chunks with rare but recurrent vocabulary. **Weakness.** A single rare entity among many common tokens is diluted by the average.

C.3. BM25

$$\text{score}_{\text{BM25}}(C) = \sum_{t \in \text{types}(C)} \text{IDF}_{\text{BM25}}(t) \cdot \frac{\text{tf}(t)\,(k_1+1)}{\text{tf}(t) + k_1\Big(1 - b + b\,\frac{|C|}{\bar{l}}\Big)}, \tag{10}$$

where $k_1 = 1.2$, $b = 0.75$, $\bar{l}$ is the mean chunk length across the corpus, and $\text{IDF}_{\text{BM25}}(t) = \log\Big(\frac{N - \text{df}(t) + 0.5}{\text{df}(t) + 0.5} + 1\Big)$.

Strengths. More robust than TF-IDF when chunk lengths vary. **Weakness.** Slightly higher compute; benefits diminish when all chunks are the same length.

C.4. Entropy (Type-Token Ratio)

$$\text{score}_{\text{Entropy}}(C) = \frac{\big|\{t : t \in C\}\big|}{|C|}. \tag{11}$$

Implemented as `(bincount(C) > 0).sum() / len(C)`.

Strengths. Corpus-independent; no IDF table required. A chunk of padding tokens or repeated stop-words scores near 0; an entity-rich chunk scores near 1. **Weakness.** Ignores global rarity; a chunk with 32 distinct common stop-words scores identically to one with 32 rare entities.

C.5. Max-IDF

$$\text{score}_{\text{Max-IDF}}(C) = \max_{p \in C} \text{IDF}(t_p). \tag{12}$$

Implemented as `idf_table[chunk_tokens].max()`.

Strengths. Acts as a pure needle detector: if any token in the chunk appears in only one block, the chunk is guaranteed to be selected. This is exactly what multi-hop tasks and NIAH require. **Weakness.** Ignores overall chunk quality; one rare token among 31 uninformative tokens still wins.

C.6. Even-Spaced (Baseline)

Selects exactly `num_chunks` × `chunk_size` tokens at uniform intervals across the block using `torch.linspace`:

$$\text{indices} = \big\lfloor \texttt{linspace}(0,\, |B| - 1,\, s) \big\rfloor, \quad s = \texttt{num\_chunks} \times \texttt{chunk\_size}. \tag{13}$$

Original (sparse) position IDs are preserved.

Strengths. No corpus statistics required; deterministic. **Weakness.** Wastes budget on low-information spans as budget grows (confirmed in ablations: `qa3` degrades from 36 to 25 as s increases from 128 to 1024).

Table 8. Summary of all scoring heuristics.

Heuristic	Signal	Corpus-dep.	Best for	Weakness
TF-IDF	Mean IDF	Yes	General-purpose	Single rare token diluted
BM25	Len-norm. IDF	Yes	Variable chunk sizes	Extra `bincount` cost
Entropy	Type-token ratio	No	Small corpus / fast	Blind to global rarity
Max-IDF	Peak IDF	Yes	Rare entity / NIAH	Ignores chunk quality
Even-Sp.	Uniform coverage	No	Coverage guarantee	Wastes budget on filler

C.7. Heuristic Summary

D. Hyperparameter Configuration

Table 9 lists all hyperparameters used in our main experiments. All values are fixed across RULER and BABILong unless noted.

Table 9. Full hyperparameter configuration.

Parameter	Value	Notes
Model		
Model	Meta-Llama-3.1-8B-Instruct	
Precision	bfloat16	
Attention impl.	FlashAttention-2	`attn_impl=flash_attention_2`
Decoding	Greedy (argmax)	Temperature = 0
Distributed setup		
Number of GPUs (B)	4	One block per GPU
Block size	25% of context length	Context split into 4 equal blocks
Pulsar Attention: summary		
Summary method	Max-IDF	Eq. (12)
Summary budget (s)	512 tokens / block(at 16k)	12.5% of block size
Chunk size	32 tokens	16 chunks selected per block
Sink size (k)	64 tokens	First 64 tokens of block 0
KV discard	Enabled	Non-B_i KV sliced after Phase 1
Position mode	Sparse	Original global position IDs preserved
Generation		
Max new tokens (BABILong)	128	
Max new tokens (RULER NIAH)	128	
Max new tokens (RULER VT)	30	
Max new tokens (RULER CWE)	120	
Max new tokens (RULER FWE)	50	
Max new tokens (RULER QA)	32	

E. RoPE Position ID Assignment

When a block's Phase 1 input is assembled as $[\text{SINK} \mid S_1, \ldots, S_{i-1} \mid B_i]$, the position IDs of the inserted sink and summary tokens can be assigned in two ways.

Sparse (default). Every token retains its *original global position* in the full context. Summary tokens from block j keep the position IDs of their source locations. The assembled sequence therefore contains *gaps* where non-selected tokens were dropped. This preserves the true inter-token distances that RoPE was trained to encode: a summary token from block 0 is still seen as far from a token in block 3, matching the model's pre-training geometry.

Contiguous. The assembled sequence is re-numbered $0, 1, \ldots, L_i - 1$ for each block i's Phase 1 forward pass (original Star Attention style). In this mode, Phase 2 query position IDs are offset to start just past the longest assembled Phase 1 length, ensuring RoPE deltas from the query to every block's KV remain small and in-distribution.

Contiguous mode can occasionally help by ensuring that summary tokens and local block tokens are seen at short relative

distances, but it distorts the global geometry. All experiments reported in this paper use **sparse** mode, which aligns with the model's pre-training distribution.

F. KV Cache Discard

After each Phase 1 forward pass, the model's KV cache contains entries for *all* tokens in the assembled input [SINK | $S_1, \ldots, S_{i-1}$ | B_i]. The sink and summary KV states are discarded immediately:

$$K_i^{(\text{layer})} \leftarrow K_i^{(\text{layer})}[:, :, \texttt{is\_bi}] \quad \text{(and same for } V_i\text{)} \tag{14}$$

where $\texttt{is\_bi}[p] = \text{True}$ iff $p \geq \texttt{summary\_len}$. Implemented via a boolean mask:

```
is_bi = torch.zeros(total_len, dtype=torch.bool)
is_bi[-block_size:] = True
kv = [[x[0][:,:,is_bi], x[1][:,:,is_bi]] for x in kv]
```

Discarding serves two purposes: **(1)** *No duplicate tokens in Phase 2*: summary token S_j was prepended to blocks $j{+}1, \ldots, B{-}1$; if retained, each KV cache would contain a copy of S_j's key/value vectors, corrupting the softmax normalisation during Phase 2's `all_gather`. **(2)** *Exact final cache size*: after discarding, each GPU holds exactly `block_size` KV entries, so the total KV cache across B GPUs equals the original context length L.

The discard can be disabled via `--no_discard_summary_kv` for experimental purposes; in that case, sink and summary KV states bake into the final cache and Phase 2 attends over them directly (though with duplicate tokens).

G. Phase 2 Distributed Softmax Merge

Phase 2 runs an identical forward pass on all B GPUs simultaneously. Each GPU computes a local attention output $\mathbf{o}_h$ and log-sum-exp ℓ_h over its own KV block. After `dist.all_gather`, the outputs are merged with the numerically stable online-softmax formula (Milakov & Gimelshein, 2018):

$$\ell' = \ell + \log\bigl(1 + e^{\ell_h - \ell}\bigr), \tag{15}$$

$$\mathbf{o}' = e^{\ell - \ell'}\,\mathbf{o} + e^{\ell_h - \ell'}\,\mathbf{o}_h, \tag{16}$$

applied iteratively over all gathered outputs. Equations (15)–(16) are implemented in `star_flash_attn/utils.py` as a `torch.jit.script` kernel for efficiency. The result is mathematically equivalent to running full attention over the concatenated KV cache of all B GPUs, at the cost of one `all_gather` communication round per generation step.