Block Sparse Flash Attention

Daniel Ohayon
Corma
Technion – Haifa, Israel
daniel@corma.ai

Itay Lamprecht*
Crusoe AI
Technion – Haifa, Israel
itayl@campus.technion.ac.il

Itay Hubara
Technion – Haifa, Israel
itayhubara@gmail.com

Israel Cohen
Technion – Haifa, Israel
icohen@ee.technion.ac.il

Daniel Soudry
Technion – Haifa, Israel
daniel.soudry@gmail.com

Noam Elata
Technion – Haifa, Israel
noamelata@technion.ac.il

Abstract

Modern large language models increasingly require long contexts for reasoning and multi-document tasks, but attention's quadratic complexity creates a severe computational bottleneck. We present *Block Sparse Flash Attention* (BSFA), a drop-in replacement that accelerates long-context inference while preserving model quality. Unlike methods that predict importance before computing scores, BSFA computes exact query-key similarities to select the top-k most important value blocks for each query. By comparing per-block maximum scores against calibrated thresholds, we skip approximately 50% of the computation and memory transfers for pruned blocks. Our training-free approach requires only a one-time threshold calibration on a small dataset to learn the per-layer and per-head attention score distributions. These distributions are stable enough across inputs that one threshold table holds the intended top-k block budget and transfers without refitting across datasets, languages, domains, and GPUs. We provide a CUDA kernel implementation that can be used as a drop-in replacement for FlashAttention. On Llama-3.1-8B, BSFA achieves up to $1.13\times$ end-to-end speedup on LongBench with only a 1.1% accuracy drop, and up to $1.24\times$ on Needle-in-a-Haystack retrieval at a 1% accuracy drop. The attention kernel itself accelerates by up to $1.38\times$. We compare BSFA against five recent sparse attention baselines (SpargeAttention, MInference, FlexPrefill, XAttention, and BLASST), and verify the method on Qwen2.5-7B and on A6000 and H100 GPUs. The implementation is available at `https://github.com/Danielohayon/Block-Sparse-Flash-Attention`.

1 Introduction

Large language models have become the backbone of modern AI systems [Brown et al., 2020, Touvron et al., 2023, Achiam et al., 2023], enabling state-of-the-art performance in generation [Achiam et al., 2023], reasoning [Wei et al., 2022, Kojima et al., 2022], and retrieval-augmented applications [Lewis et al., 2020, Borgeaud et al., 2022]. At their core lies scaled dot-product attention [Vaswani et al., 2017], whose quadratic compute and storage in the sequence length becomes the dominant bottleneck as practitioners push contexts to hundreds of thousands of tokens, where attention operations consume the majority of inference time and can account for over 90% of total FLOPs in long-context scenarios. FlashAttention [Dao et al., 2022, Dao, 2023] eliminates the memory side of this bottleneck by tiling computation into streaming updates, but the arithmetic work of multiplying every query against every key, applying softmax, then aggregating values remains $O(N^2)$ FLOPs, a cost that in general cannot be made subquadratic [Alman and Song, 2023, Alman and Yu, 2024].

*Work done while at Habana Labs (Intel).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

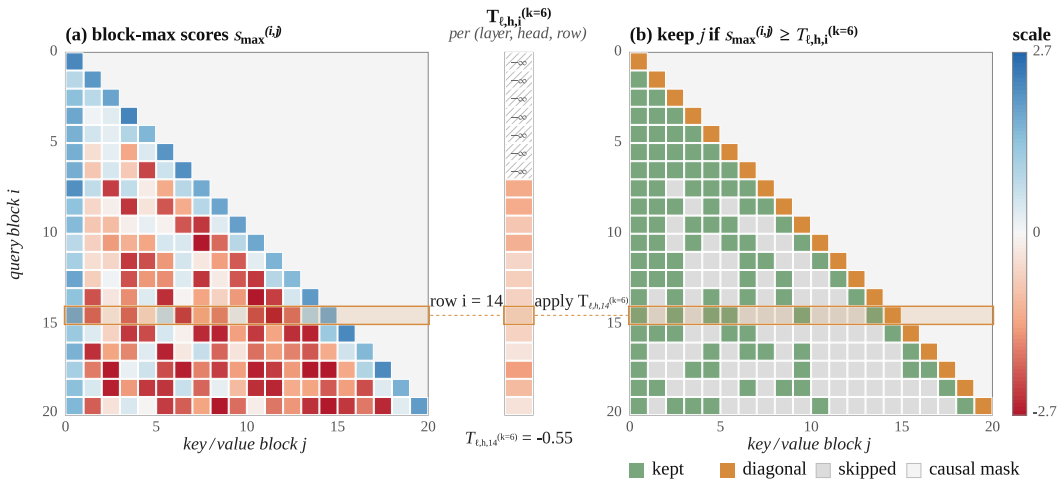

Figure 1: BSFA mechanism. **(a)** Per-block maxima $s_{\max}^{(i,j)}$ are computed exactly within the FlashAttention tile. The center column visualizes the calibrated threshold tensor $T_{\ell,h,i}^{(k)}$ ($k=6$ shown), one value per query row, with hatched $-\infty$ entries where fewer than k off-diagonal candidates exist. **(b)** Off-diagonal blocks are kept (green) if $s_{\max}^{(i,j)} \geq T_{\ell,h,i}^{(k)}$. Diagonal blocks (orange) always run; skipped blocks (grey) avoid the $\mathbf{V}_j$ load and the $\mathbf{PV}$ matmul. The highlighted row $i=14$ traces a single gating decision across all three panels.

Empirical evidence shows that attention distributions are highly sparse (Section 2), motivating sparse attention methods that reduce computation by avoiding certain query-key interactions [Beltagy et al., 2020, Roy et al., 2020, Kitaev et al., 2020]. The dominant strategy in this literature is to predict importance *before* observing the scores (via fixed patterns, hashing, clustering, or pooled approximations of $\mathbf{QK}^\top$) and commit to pruning decisions on that prediction. Such predictive early pruning risks accuracy degradation when relevant tokens are scattered unpredictably across long contexts.

We propose Block Sparse Flash Attention (BSFA), which inverts this strategy. BSFA computes all query-key scores exactly within FlashAttention's tiled framework, then uses the per-block maximum scores to decide which value blocks to process. Blocks whose maximum scores fall below a calibrated threshold are skipped, so their values are not loaded from HBM, the $\mathbf{PV}$ multiplication is not performed, and the running softmax statistics are not updated. This exploits the observation that blocks with uniformly low scores contribute negligibly after softmax normalization. If no query-key pair in a block has high enough similarity, the entire block can be skipped without meaningful impact on the output. By preserving exact score computation while selectively skipping the value-side work, BSFA achieves substantial speedups without the accuracy risks of early pruning.

Across our evaluation, BSFA is the only method that reaches the $\geq$99%-accuracy band on LongBench while accelerating long-context inference and avoiding the slowdowns at short and medium context lengths that affect MInference, FlexPrefill, and XAttention. Experiments on Llama-3.1-8B demonstrate that BSFA achieves up to $1.13\times$ end-to-end speedup on real-world LongBench tasks with only a 1.1% drop from baseline accuracy, and up to $1.24\times$ on Needle-in-a-Haystack retrieval at 99% baseline accuracy. The underlying attention kernel itself is sped up by up to $1.38\times$ at higher sparsity levels (Section 5). Comparing against five recent sparse attention methods (SpargeAttention [Zhang et al., 2025], MInference [Jiang et al., 2024], FlexPrefill [Lai et al., 2025], XAttention [Xu et al., 2025], and BLASST [Yuan et al., 2025]), BSFA reaches the $\geq$99%-accuracy band on LongBench at speedups comparable to MInference and XAttention, while SpargeAttention and FlexPrefill cannot reach this band and BLASST loses more accuracy at matched sparsity. The pre-attention selection methods (MInference, FlexPrefill, XAttention) additionally regress to as low as $0.17\times$ on contexts under 16K, where BSFA stays within a few percent of dense. We additionally verify these results on Qwen2.5-7B and on A6000 and H100 GPUs. We release a CUDA kernel extending FlashAttention-2 as a drop-in replacement.

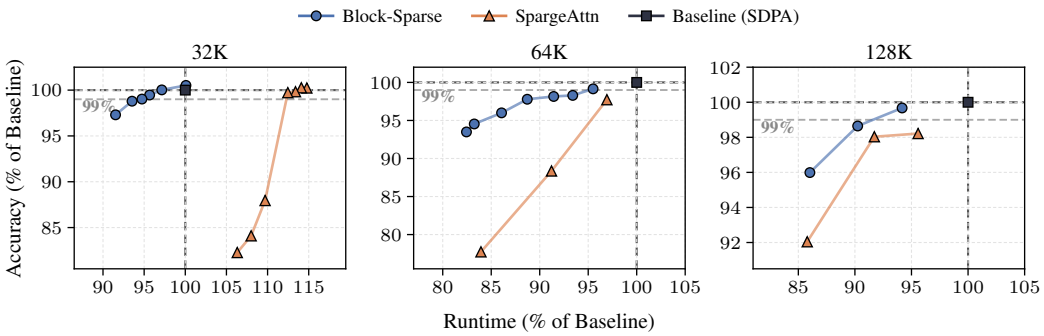

Figure 2: Accuracy-latency trade-offs on RULER for 32K (left), 64K (center), and 128K (right). BSFA (blue) maintains high accuracy with consistent speedups, outperforming SpargeAttention (orange). Each point is a different sparsity level (Table 6).

2 Background

Scaled dot-product attention. Transformers use multi-head scaled dot-product attention to process sequences [Vaswani et al., 2017], where each head computes $\text{softmax}(\mathbf{QK}^\top/\sqrt{d})\mathbf{V}$ and $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are linear projections of the input. Both the score computation $\mathbf{QK}^\top$ and value aggregation $\mathbf{PV}$ scale as $O(N^2 d)$, dominating computation for long sequences. For Llama-3.1-8B at 128K tokens each of these quadratic operations alone requires $\sim$32$\times$ the FLOPs of a single linear projection (see Appendix B.1).

FlashAttention FlashAttention-2 (FA-2) [Dao, 2023] computes exact attention without materializing the full $N \times N$ score matrix by tiling queries into M_Q blocks of size B_M and keys/values into M_{KV} blocks of size B_N, keeping intermediate results in fast on-chip SRAM while streaming from HBM. For each tile (i, j), FA-2 performs two equally expensive matrix multiplications (computing scores $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top$ and the value aggregation $\tilde{\mathbf{P}}_{ij}\mathbf{V}_j$, each $2B_M B_N d$ FLOPs) while loading both $\mathbf{K}_j$ and $\mathbf{V}_j$ blocks from HBM. FA-2 removes the memory bottleneck but still computes all $O(N^2)$ interactions, motivating block-sparse approaches that selectively skip negligible blocks (Appendix B.2 provides full algorithmic details).

Empirical sparsity patterns. Attention distributions in transformers exhibit inherent sparsity. The phenomenon of *rank collapse* [Dong et al., 2021, Saada et al., 2025] forces attention matrices to degenerate into low-rank forms in deeper layers, concentrating mass on a small number of positions. As a result, models consistently develop *attention sinks* at initial positions [Xiao et al., 2024] and concentrate on sparse "heavy hitter" tokens [Zhang et al., 2023, Zucchet et al., 2025], while most query-key interactions contribute negligibly to the output [Tang et al., 2024].

3 Related Work

As transformers scale to longer contexts, various approaches have emerged to reduce attention's quadratic complexity. Since attention consists of three main operations (computing QK scores, applying softmax, and computing PV products), different methods target different subsets of these operations. We categorize existing work by which operations they optimize and when sparsity decisions are made.

Predicting importance before scoring. Methods that avoid computing $\mathbf{QK}^\top$ for certain pairs can in principle achieve the greatest speedup by eliminating all three operations for pruned blocks, but must predict importance without observing the actual attention scores. *Pattern-based* approaches fix sparsity a priori using local windows with global tokens [Beltagy et al., 2020], strided patterns [Child et al., 2019], or block-diagonal structures [Gupta et al., 2021], and cannot adapt to content-specific importance. *Content-aware* approaches predict importance from approximations of the score matrix. Hashing groups similar tokens [Kitaev et al., 2020], clustering does the same with online k-means [Roy et al., 2020], and Quest [Tang et al., 2024] uses lightweight query-aware heuristics to prune key-value pairs.

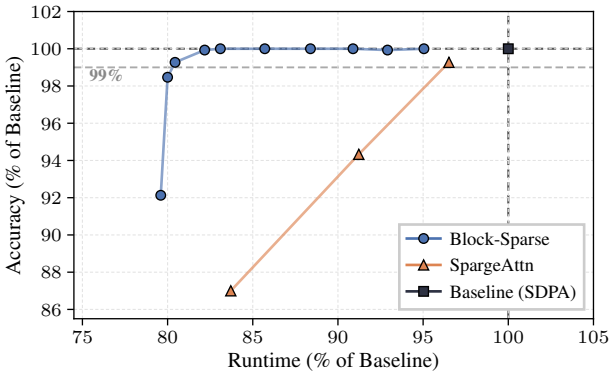

Figure 3: Accuracy vs. Time-to-First-Token (TTFT) speedup on the 64K Needle-in-a-Haystack Single-key task. BSFA retains 99% of baseline accuracy at k=32 with a $1.24\times$ speedup, exploiting that retrieval concentrates signal in few blocks. SpargeAttention is shown as a multi-operating-point reference.

A more recent line of work targets prefill specifically. SpargeAttention [Zhang et al., 2025] compresses Q/K blocks via mean-pooling for a coarse global filter, and inside the kernel additionally skips $\mathbf{PV}$ when the block local maximum is far below the running global maximum. MInference [Jiang et al., 2024] assigns each attention head offline to one of three sparse patterns (A-shape, Vertical-Slash, Block-Sparse) via kernel-aware search. FlexPrefill [Lai et al., 2025] chooses per-head patterns and indices online from pooled-block and last-block approximations, controlled by a coverage threshold γ. XAttention [Xu et al., 2025] sums strided antidiagonals of the score block as a cheap importance proxy, selecting blocks by a cumulative-mass threshold. What unifies these methods is that pruning commits before exact scores are seen, with sparse patterns varying across queries, heads, and inputs. Variable per-query workload bottlenecks GPU kernels at the slowest thread, and the pre-attention work introduces a fixed overhead that becomes a regression on shorter sequences, a phenomenon we observe in Section 5.

Gating after scoring. H2O [Zhang et al., 2023] tracks cumulative attention weights during decoding and evicts low-scoring KV cache entries. This targets decode-time memory rather than prefill compute, and evicted tokens cannot be recovered later. StreamingLLM [Xiao et al., 2024] retains attention-sink tokens plus a sliding window for streaming, but uses fixed patterns rather than content-based selection. KV-cache compression methods such as KVzip [Kim et al., 2025] and KVzap [Jegou and Jeblick, 2026] reduce the memory footprint of the KV cache. Neither reports a prefill or time-to-first-token speedup, the latency BSFA targets, so they complement BSFA rather than replace it.

BLASST [Yuan et al., 2025], like BSFA, gates blocks based on attention-score maxima rather than approximations, and is the closest method to ours methodologically. The two approaches differ in threshold parameterization and in comparison reference. BLASST applies a single global threshold per (model, target-sparsity) scaling with context length, while BSFA stores a per-(layer, head, position) threshold table calibrated offline that directly targets the per-layer/head heterogeneity shown in BLASST's Figure 7. BLASST also gates based on the gap to a *running* global maximum (variable sparsity across queries and positions), where BSFA's fixed-k design yields a uniform per-query workload that avoids the slowest-thread bottleneck. We compare against BLASST's public release on H100, since no A100 implementation is provided.

Δ-Attention [Willette et al., 2025] addresses the complementary problem of correcting distributional shift from sparse prefill via dense-sparse output differences for sampled query rows. This correction is orthogonal to block selection and could potentially be combined with BSFA.

Learned sparsity. SparseK [Lou et al., 2024] and SeerAttention [Gao et al., 2024b] learn which keys to attend to through trainable scoring networks with differentiable selection. While these yield adaptive, content-aware sparsity, they require modifying the model architecture and fine-tuning. In contrast, our method remains training-free: we calibrate thresholds offline on a small dataset without updating any model weights or introducing learnable components.

Approximation methods. Orthogonal to selection-based sparsity, kernel-based [Choromanski et al., 2020] and low-rank [Xiong et al., 2021] methods replace softmax attention with computationally cheaper approximations. While these offer strong asymptotic complexity guarantees, they fundamentally alter the attention mechanism and require architectural changes.

4 Method

We present Block Sparse Flash Attention (BSFA), which accelerates long-context prefill by selecting the top-k most important value blocks for each query using calibrated thresholds (Figure 1). Building on FlashAttention's [Dao, 2023] tiled computation, BSFA adds a single check after each score block: if the per-block maximum score falls below a calibrated threshold specific to that layer, head, and position, the block's values are not loaded from HBM and the corresponding $\mathbf{PV}$ multiplication is skipped.

Block-level sparsity can be reliably detected through exact-score maxima rather than predicted before observing scores. Computing all $\mathbf{QK}^\top$ products preserves attention fidelity, while the skipped $\mathbf{PV}$ multiplication and value loads constitute approximately 50% of attention FLOPs and 50% of HBM bandwidth (Section 4.3). Operating at block granularity aligns with tensor cores and amortizes each gating decision over thousands of value loads. Following FlashAttention's tiling, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ are partitioned into blocks of size B_M and B_N, and the number of retained blocks k serves as our sparsity parameter. An additional advantage of fixed-k is improved GPU utilization, since variable sparsity creates workload imbalance bottlenecked by the slowest thread.

4.1 Gating Mechanism

Our gating mechanism determines which value blocks to process by examining the maximum attention score within each query-key block pair. For each block pair (i, j) with $j < i$ in the causal setting, we compute the standard scaled dot-product scores (as defined in Section 2):

$$\mathbf{S}_{ij} = \frac{\mathbf{Q}_i \mathbf{K}_j^\top}{\sqrt{d}} \in \mathbb{R}^{B_M \times B_N} \tag{1}$$

and extract the maximum score:

$$s_{\max}^{(i,j)} = \max_{p,q} [\mathbf{S}_{ij}]_{pq} \tag{2}$$

We select the top-k blocks per query using thresholds calibrated per (layer, head, position). Unlike SpargeAttention, which skips blocks whose post-softmax contributions become numerically negligible after exponential scaling, BSFA targets a fixed number of retained blocks for consistent workload across inputs. We gate each block by comparing $s_{\max}^{(i,j)}$ against $T_{\ell,h,i}^{(k)}$ (Figure 1b). If below, the block is skipped entirely: values are not loaded, the $\mathbf{PV}$ multiplication is not performed, and running statistics are not updated. The threshold parameterization over (ℓ, h, i, k) captures attention's empirical heterogeneity across layers (local in lower, sink-attentive in deeper), heads (local vs. global specialists), and positions [Xiao et al., 2024]. Diagonal blocks bypass the gate (Figure 1b, orange) to preserve causal correctness and local context.

4.2 Threshold Calibration

A key observation underlying our approach is that attention patterns remain relatively consistent across different inputs for a given model, allowing offline calibration on a small dataset (Figure 4). The procedure is simple: for each query position, we sort off-diagonal blocks by their maximum scores and select the threshold that retains exactly the top-k blocks, averaging across calibration samples for robustness. We store thresholds per layer, head, and position ($\mathbf{T} \in \mathbb{R}^{L \times H \times \lceil N_{\max}/B_M \rceil}$), with multiple sparsity levels enabling dynamic accuracy-speed trade-offs at deployment without recalibration (see Appendix B.3 for details).

4.3 Complexity Analysis

By skipping blocks with low maximum scores, BSFA eliminates substantial computation and memory traffic. For each skipped block, we avoid:

Compute operations: The dominant saving comes from skipping the $\mathbf{PV}$ multiplication, which requires $2B_M B_N d$ FLOPs per block, the same cost as the $\mathbf{QK}$ computation. We also skip exponential computations for softmax ($B_M B_N$ operations), running statistics updates ($O(B_M d)$ operations), and normalizer maintenance ($O(B_M B_N)$ operations).

Memory bandwidth: We avoid loading value blocks from HBM, saving $B_N d$ elements per skipped block. Since keys and values are the same size, this represents exactly 50% of the key-value memory traffic.

The overhead of computing block maxima and threshold comparisons is negligible compared to the tensor operations we skip.

Theoretical speedup bound and kernel efficiency. The attention speedup BSFA can reach has a simple theoretical bound. Per tile, the score half ($\mathbf{Q}_i\mathbf{K}_j^\top$ and the $\mathbf{K}_j$ load) and the value half ($\tilde{\mathbf{P}}_{ij}\mathbf{V}_j$ and the $\mathbf{V}_j$ load) cost the same. BSFA always runs the score half and runs the value half for a fraction ρ of the causal tiles, the retained block density of Eq. 3. It therefore does $1+\rho$ units of work where dense attention does 2, which bounds the attention-kernel speedup by $2/(1+\rho)$, on arithmetic and on HBM traffic alike. The bound does not depend on the sequence length N, because both halves grow identically with N, and skipping only value-side work can never exceed $2\times$. We define the kernel's implementation efficiency as its measured speedup divided by this bound. On RULER 64K, our kernel reaches $1.38\times$, $1.30\times$, and $1.19\times$ at k=192, 256, and 384 (Table 3), where the measured densities are 0.36, 0.45, and 0.61 (Table 6). Its implementation efficiency is therefore 94%, 94%, and 96% of the theoretical bound, so our kernel is very close to optimal.

4.4 Algorithm and Implementation

Algorithm 1 presents our block sparse Flash Attention for the causal prefill case. The modification to standard FlashAttention is minimal: we add a gating check after computing each score block but before loading values. The integration requires only a conditional branch in the existing kernel: if $s_{\max}^{(i,j)} < T_{\ell,h,i}^{(k)}$ (where the threshold is specific to layer ℓ, head h, and position i), we skip value loading and all subsequent operations for that block. Skipped blocks do not participate in the running statistics $(m_i, \ell_i, \mathbf{O}_i)$ maintained by the streaming softmax algorithm.

Algorithm 1 Block Sparse Flash Attention (simplified for single query block i, causal)

Input: Query block $\mathbf{Q}_i \in \mathbb{R}^{B_M \times d}$, Key blocks $\{\mathbf{K}_j\}$, Value blocks $\{\mathbf{V}_j\}$, Thresholds $T_{\ell,h,i}^{(k)}$
Initialize: $m_i = -\infty \cdot \mathbf{1}_{B_M}$, $\ell_i = \mathbf{0}_{B_M}$, $\tilde{\mathbf{O}}_i = \mathbf{0}_{B_M \times d}$ {un-scaled accumulator}
Load $\mathbf{Q}_i$ from HBM to SRAM
for each key/value block $j = 0, \ldots, i-1$ **do**
  Load $\mathbf{K}_j$ from HBM to SRAM
  Compute scores: $\mathbf{S}_{ij} = \mathbf{Q}_i\mathbf{K}_j^\top / \sqrt{d}$ {in SRAM}
  Compute block maximum: $s_{\max}^{(i,j)} = \max(\mathbf{S}_{ij})$
  if $s_{\max}^{(i,j)} > T_{\ell,h,i}^{(k)}$ **then**
    Update row-wise max: $m_i^{\text{new}} = \max(m_i, \text{rowmax}(\mathbf{S}_{ij}))$
    Compute $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - m_i^{\text{new}})$ {stable softmax}
    Load $\mathbf{V}_j$ from HBM to SRAM
    Rescale accumulator: $\tilde{\mathbf{O}}_i \leftarrow \tilde{\mathbf{O}}_i \cdot \text{diag}(\exp(m_i - m_i^{\text{new}}))$
    Update un-scaled output: $\tilde{\mathbf{O}}_i \leftarrow \tilde{\mathbf{O}}_i + \tilde{\mathbf{P}}_{ij}\mathbf{V}_j$
    Update normalizer: $\ell_i \leftarrow \ell_i \cdot \exp(m_i - m_i^{\text{new}}) + \text{rowsum}(\tilde{\mathbf{P}}_{ij})$
    $m_i \leftarrow m_i^{\text{new}}$
  end if
end for
Process diagonal block $j = i$ with causal mask applied to $\mathbf{S}_{ii}$ {always process, no gating}
Finalize: $\mathbf{O}_i \leftarrow \tilde{\mathbf{O}}_i \oslash \ell_i$ {final scaling, element-wise division}
Store $L_i = m_i + \log(\ell_i)$ {logsumexp for backward pass}
Return: $\mathbf{O}_i$

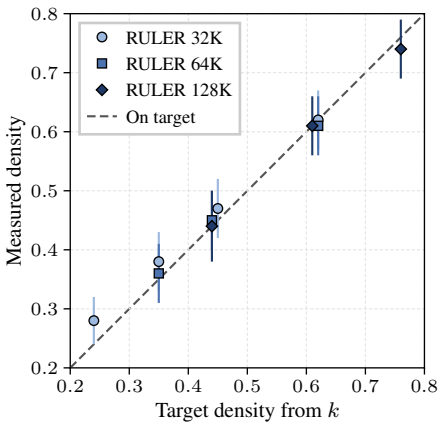

Figure 4: Measured block density against the target set by k (Eq. 3) for every BSFA configuration on RULER 32K, 64K, and 128K, with bars of one standard deviation across samples (Table 6). Thresholds calibrated once, offline, hold their block budget at inference at every length, with a standard deviation of at most 0.06.

5 Experimental Evaluation

We evaluate Block Sparse Flash Attention (BSFA) on diverse long-context benchmarks to demonstrate its effectiveness in reducing prefill latency while maintaining model quality. Our experiments span multiple sequence lengths, two language-model architectures, three GPU architectures, and five recent sparse-attention baselines.

5.1 Experimental Setup

Hardware and implementation. Our primary experiments are conducted on NVIDIA A100 80GB GPUs with CUDA 12.1. We implemented BSFA as a custom CUDA kernel extending FlashAttention 2 [Dao, 2023], with block dimensions $B_M = 128$ and $B_N = 64$ optimized for A100's tensor cores. The kernel modifications add threshold-based gating logic with minimal overhead. BSFA uses FP16 precision throughout, maintaining numerical fidelity while achieving speedups through selective computation. To verify generality, we additionally evaluate on RTX A6000 (Ampere, lower memory bandwidth) and on H100 with a Hopper-compatible kernel port. Details are deferred to the generality discussion in §5 and Appendix A.3. Our complete implementation is available at `https://github.com/Danielohayon/Block-Sparse-Flash-Attention`, and all reported results are fully reproducible using the provided scripts.

Models and datasets. We primarily evaluate on Llama-3.1-8B-Instruct [Grattafiori et al., 2024], supplementing with cross-model evaluation on Qwen2.5-7B-Instruct [Team, 2025] (28 layers, 28 heads, 4 KV heads, GQA-7) to verify that the method generalizes to a different architecture. We use three benchmarks. **LongBench** [Bai et al., 2024] is a real-world long-document understanding benchmark spanning multiple domains. We report accuracy on the full 27-task benchmark ($\sim$4,750 samples) and TTFT on two subsets, the 10 longest `narrativeqa` samples ($\sim$65K tokens each) and a mixed-length subset of 160 samples spanning 295–65,461 tokens for the variable-length analysis. **RULER** [Hsieh et al., 2024] provides synthetic tasks testing precise retrieval and reasoning over long contexts (32K, 64K, 128K tokens). **Needle-in-a-Haystack** provides targeted retrieval at 64K.

Crucially, the calibration and evaluation datasets are disjoint across task categories. RULER contains multiple task categories (Retrieval, Multi-hop Tracing, Aggregation, Question Answering, etc.), which we split into two groups. For threshold calibration, we use 16 samples from the first group of categories. For RULER latency measurement, we select samples per sequence length exclusively from the held-out categories not used during calibration, ensuring our speedup measurements reflect generalization to unseen task types. TTFT exhibits low variance across samples of similar length, making 10 samples sufficient for reliable timing in our primary measurements. Per-experiment sample counts are noted in their respective tables. We use the same RULER-calibrated thresholds for our LongBench evaluation (Table 1), demonstrating strong cross-dataset generalization without any

Table 1: Main results on LongBench (Llama-3.1-8B). Accuracy is on the full 27-task benchmark (∼4,750 samples) via lm-evaluation-harness, and speedup is end-to-end TTFT on the 10 longest `narrativeqa` samples (∼65K tokens each). All methods on A100 unless marked ⋆ (H100, since BLASST has no public A100 implementation). Sliding Window does not run as a sparse-attention kernel against the same baseline, so we report accuracy only. Bracketed numbers show relative degradation from the dense baseline.

Method	Param.	Accuracy ↑	Speedup ↑
Dense FlashAttention-2	All	40.24%	1.00×
BSFA	k=32	39.39% (-2.1%)	1.16×
	k=64	39.78% (-1.1%)	1.13×
	k=96	39.88% (-0.9%)	1.10×
	k=128	40.03% (-0.5%)	1.08×
SpargeAttention	τ=0.4	30.06% (-25.3%)	1.09×
	τ=0.5	33.11% (-17.7%)	1.02×
	τ=0.6	35.87% (-10.9%)	0.98×
MInference	default	40.06% (-0.45%)	1.17×
FlexPrefill	γ=0.9	35.11% (-12.8%)	1.59×
	γ=0.99	38.90% (-3.3%)	1.06×
XAttention	default	39.95% (-0.7%)	1.16×
BLASST⋆	50%	39.23% (-2.5%)	1.11×
	70%	36.40% (-9.5%)	1.16×
Sliding Window	64 (4,096 tokens)	13.43% (-66.6%)	—
	96 (6,144 tokens)	20.29% (-49.6%)	—

dataset-specific tuning. For accuracy evaluation, we test the full RULER and LongBench benchmarks using the lm-evaluation-harness framework [Gao et al., 2024a].

Baselines. Our exact-attention reference is **Dense FlashAttention-2** [Dao, 2023] (we use FA-2 since FA-3 is optimized for Hopper). On the sparse-attention side we evaluate **SpargeAttention** [Zhang et al., 2025], **MInference** [Jiang et al., 2024], **FlexPrefill** [Lai et al., 2025], **XAttention** [Xu et al., 2025], and **BLASST** [Yuan et al., 2025], all described in Section 3. We follow each method's recommended configuration and sweep their natural accuracy-speed knobs (SpargeAttention's τ, FlexPrefill's γ, BLASST's sparsity target). Since BLASST has no A100 release, we run it against their public H100 release. We additionally include a **Sliding Window** baseline at the same block budget as BSFA, to confirm content-aware selection does not simply collapse to a local pattern. Method-specific implementation choices (kernel selection, dtype constraints, and other artifact-specific notes) are documented in Appendix A.5.

Metrics. We measure three quantities. **End-to-end TTFT** is the wall-clock prefill time, averaged over 10 runs after warmup. **Attention-kernel-only TTFT** isolates the attention call via CUDA-event timing around the kernel boundary, allowing us to disambiguate kernel-level speedup from end-to-end speedup that includes work BSFA cannot accelerate (Q/K/V/O projections, RoPE, GQA expansion, MLP, the LM head). **Task accuracy** is evaluated using lm-evaluation-harness [Gao et al., 2024a]. We additionally validate threshold effectiveness by comparing predicted versus measured block density. For query block i,

$$\rho_i = \frac{\min(k, i) + 1}{i + 1}, \tag{3}$$

where k is the number of retained off-diagonal blocks and $+1$ accounts for the always-retained diagonal block. Measured density is the actual fraction retained on validation data. We measure both accuracy and latency on the *same* sequence lengths and samples for each sparsity configuration.

5.2 Main Results

LongBench accuracy and speedup. Table 1 presents the head-to-head comparison. BSFA's k slider exposes a flexible operating range from $1.08\times/-0.5\%$ (k=128) to $1.16\times/-2.1\%$ (k=32), all served from a single calibrated threshold tensor. MInference operates at a fixed point ($1.17\times/-0.45\%$)

Table 2: End-to-end TTFT speedup on a LongBench mixed-length subset (160 samples spanning 295–65,461 tokens), broken down by sequence-length bucket. Speedup is reported against dense FlashAttention-2 within each bucket. Accuracy column is the full 27-task LongBench score from Table 1, repeated here for ease of speedup-vs-accuracy correlation. Llama-3.1-8B on A100 unless marked $\star$ (H100). SpargeAttention was not measured on this subset. **Bold** cells highlight regressions below dense ($<1.00\times$).

Method	Param.	Accuracy ↑	<4K	4–8K	8–16K	16–32K	32–65K
Dense FlashAttention-2	All	40.24%	1.00×	1.00×	1.00×	1.00×	1.00×
BSFA	k=64	39.78% (-1.1%)	**0.96**×	**0.97**×	1.01×	1.05×	1.19×
MInference	default	40.06% (-0.45%)	**0.17**×	**0.31**×	**0.48**×	**0.58**×	**0.99**×
FlexPrefill	γ=0.9	35.11% (-12.8%)	**0.71**×	**0.90**×	1.05×	1.18×	1.59×
	γ=0.99	38.90% (-3.3%)	**0.69**×	**0.80**×	**0.93**×	**0.99**×	1.12×
XAttention	default	39.95% (-0.7%)	**0.85**×	**0.90**×	**0.97**×	1.01×	1.09×
BLASST*	50%	39.23% (-2.5%)	**0.97**×	1.01×	1.03×	1.06×	1.08×
	70%	36.40% (-9.5%)	1.00×	1.02×	1.05×	1.09×	1.11×

chosen offline by pattern search, with no accuracy-speed knob and a per-task spread from -6.0% on hotpotqa to $+9.3\%$ on passage_count, plus a pre-attention overhead regression to $0.17\times$ at <4K (Table 2). XAttention reaches $1.16\times$ at -0.7% with a milder overhead floor ($0.85\times$ at <4K). BLASST trades accuracy more aggressively at matched speedup, reaching $1.11\times$ at 50% sparsity but losing 2.5% versus BSFA k=64 at $1.13\times$ with only 1.1% loss. SpargeAttention loses at least 10.9% and FlexPrefill at its best-quality γ=0.99 still loses 3.3% at $1.06\times$, neither reaching the 99%-accuracy band. The sliding-window baseline at the same block budget collapses to $\leq 21\%$ accuracy, confirming that content-aware selection is essential.

Variable-length analysis. Table 2 reports per-bucket TTFT on the mixed-length subset. Pre-attention selection methods incur a fixed overhead per call and regress on shorter contexts (MInference $0.17\times$ below 4K, FlexPrefill γ=0.99 $0.69\times$, XAttention $0.85\times$). BSFA's gate is one block-max comparison inline in the FA-2 kernel with essentially no overhead, mildly negative ($\sim 0.96\times$) below 8K and positive at $\geq$8K. BLASST on H100 is Hopper-only and loses more accuracy at matched sparsity (Table 1).

RULER 64K and the full sweep. BSFA's k provides a continuous accuracy-speedup knob. Figure 2 traces this curve against SpargeAttention at RULER 32K, 64K, and 128K, with BSFA outperforming SpargeAttention's full sweep at every context length. The same calibrated thresholds let BSFA cover the high-accuracy regime on LongBench (k=96 at $1.10\times$/99.1%, Table 1), an operating point FlexPrefill (-3.3%) and BLASST (-2.5%) cannot reach at any evaluated configuration. At more aggressive sparsity, BSFA's k=192 on RULER 64K is competitive in speedup with MInference, FlexPrefill at γ=0.99, and BLASST 50% on H100 (full comparison in Appendix Table 7).

Attention-only versus end-to-end speedup. End-to-end TTFT includes work BSFA cannot accelerate, such as Q/K/V/O projections, RoPE, GQA expansion, MLP, and the LM head. Table 3 isolates the attention kernel via CUDA-event timing around the kernel boundary. At RULER 64K k=192, the attention kernel itself is $1.38\times$ faster, and amortizes to $1.13\times$ over the full forward pass. The kernel-only number is the apples-to-apples FlashAttention-replacement metric, while the end-to-end number is what users observe when BSFA is dropped into a full inference pipeline.

Table 3: Attention-kernel-only versus end-to-end speedup at RULER 64K (BSFA, Llama-3.1-8B, A100). Accuracy is on RULER 64K with relative degradation from dense in brackets. Kernel-only timing isolates the attention call from work BSFA cannot accelerate (Q/K/V/O projections, RoPE, GQA expansion, MLP, the LM head).

Param.	Accuracy ↑	E2E Speedup ↑	Attn-only Speedup ↑
SDPA	84.96%	1.00×	1.00×
k=192	83.08% (-2.2%)	1.13×	**1.38**×
k=256	83.39% (-1.8%)	1.09×	1.30×
k=384	84.24% (-0.8%)	1.05×	1.19×

Task-adaptive sparsity (Needle-in-a-Haystack). BSFA's content-aware sparsity particularly excels at targeted retrieval. Figure 3 presents the 64K Needle-in-a-Haystack Single-key task, where BSFA maintains 99% of baseline accuracy with only k=32 blocks at $1.24\times$ speedup, a much smaller block budget than the high-accuracy LongBench operating points ($k \geq 96$) in Table 1. The disparity is task-adaptive. Needle-finding concentrates critical information in a few specific locations that exact-score gating reliably identifies, while haystack distractors receive uniformly low scores. Tasks requiring broader context (summarization, multi-hop reasoning) need more blocks to capture distributed information.

Table 4: BSFA on additional GPU architectures (Llama-3.1-8B). A6000 reuses A100-calibrated thresholds *without recalibration* on RULER 64K. H100 uses a Hopper-compatible kernel against an FA-2-style baseline on LongBench longest-10.

GPU	k=32	k=64	k=128
A6000	1.31×	1.28×	1.23×
H100	1.25×	1.21×	1.12×

Table 5: BSFA on Qwen2.5-7B (A100, GQA-7, 28L/28H/4 KV-heads, separately calibrated thresholds). Accuracy is on full LongBench, and speedup is end-to-end TTFT on RULER 64K.

Param.	Accuracy ↑	Speedup ↑
Dense FA-2	34.05%	1.00×
k=16	33.17% (-2.6%)	1.17×
k=32	33.80% (-0.7%)	1.18×
k=64	34.06% (+0.0%)	1.18×

Generalization across hardware, models, and datasets. Tables 4 and 5 confirm BSFA transfers to A6000 (with A100-calibrated thresholds reused *without recalibration*), to H100 (FA-2-style port), and to Qwen2.5-7B (matching its dense LongBench baseline at k=64). RULER-calibrated thresholds also drive our LongBench results (Table 1, including the $\sim$12% Chinese portion), with measured density tracking Eq. 3 (std ≤ 0.06, Appendix Table 6). Across these benchmarks, BSFA uniquely satisfies both targets, $\geq$99% LongBench accuracy and $\geq 0.96\times$ end-to-end speedup at every length bucket. MInference, FlexPrefill, and XAttention regress on contexts below 16K, and BLASST loses more accuracy at matched sparsity.

Limitations and conclusion. BSFA targets prefill (decode is complementary [Zhang et al., 2023, Xiao et al., 2024]) and extends FA-2, leaving FA-3 integration to future work. Block granularity keeps a whole block for a single important token, which limits sparsity where attention is diffuse. It is also a strength, since a block is skipped only when all of its scores fall below threshold, so BSFA never drops a high-scoring token and always keeps the context around it. BSFA is a training-free FA-2 replacement using calibrated per-(layer,head,position) thresholds on exact scores, validated across datasets, languages, models (Llama-3.1, Qwen2.5), and GPUs (A100, A6000, H100).

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their feedback and Habana Labs (Intel) for their support during the early stages of this work. This research was supported by the Israel Science Foundation (grant no. 1449/23) and the Pazy Research Foundation.

References

Josh Achiam et al. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Josh Alman and Zhao Song. Fast attention requires bounded entries, 2023. URL https://arxiv.org/abs/2302.13214.

Josh Alman and Hantao Yu. Fundamental limitations on subquadratic alternatives to transformers, 2024. URL https://arxiv.org/abs/2410.04271.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens, 2022. URL https://arxiv.org/abs/2112.04426.

Tom B. Brown et al. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020. URL https://arxiv.org/abs/2009.14794.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth, 2021. URL https://arxiv.org/abs/2103.03404.

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

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning intrinsic sparse attention in your LLMs, 2024b. URL https://arxiv.org/abs/2410.13276.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention, 2021. URL https://arxiv.org/abs/2106.06899.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Simon Jegou and Maximilian Jeblick. KVzap: Fast, adaptive, and faithful KV cache pruning, 2026. URL https://arxiv.org/abs/2601.07891.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention, 2024. URL https://arxiv.org/abs/2407.02490.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction, 2025. URL https://arxiv.org/abs/2505.23416.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. URL https://arxiv.org/abs/2001.04451.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2022. URL https://arxiv.org/abs/2205.11916.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference, 2025. URL https://arxiv.org/abs/2502.20766.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks, 2020. URL https://arxiv.org/abs/2005.11401.

Chao Lou, Zixia Jia, Zilong Zheng, and Kewei Tu. Sparser is faster and less is more: Efficient sparse attention for long-range transformers, 2024. URL https://arxiv.org/abs/2406.16747.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers, 2020. URL https://arxiv.org/abs/2003.05997.

Thiziri Nait Saada, Alireza Naderi, and Jared Tanner. Mind the gap: a spectral analysis of rank collapse and signal propagation in attention layers. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sRKtbGsebH.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference, 2024. URL https://arxiv.org/abs/2406.10774.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Hugo Touvron et al. LLaMA: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL https://arxiv.org/abs/2201.11903.

Jeffrey Willette, Heejun Lee, and Sung Ju Hwang. Delta attention: Fast and accurate sparse attention inference by delta correction. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2025. URL https://arxiv.org/abs/2505.11254.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention, 2021. URL https://arxiv.org/abs/2102.03902.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block sparse attention with antidiagonal scoring, 2025. URL https://arxiv.org/abs/2503.16428.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Richard Cai, Julien Demouth, John D. Owens, Xia Hu, Song Han, Timmy Liu, and Huizi Mao. BLASST: Dynamic blocked attention sparsity via softmax thresholding, 2025. URL https://arxiv.org/abs/2512.12087.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference, 2025. URL https://arxiv.org/abs/2502.18137.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Nicolas Zucchet, Francesco d'Angelo, Andrew K. Lampinen, and Stephanie C. Y. Chan. The emergence of sparse attention: impact of data distribution and benefits of repetition, 2025. URL https://arxiv.org/abs/2505.17863.

A Additional Experimental Results

A.1 Full RULER Results

Table 6 reports accuracy, predicted/measured block density, and end-to-end TTFT speedup for BSFA, SpargeAttention, and Dense FlashAttention-2 across all three RULER lengths (32K, 64K, 128K). The 64K speedup column is the source for the BSFA and Sparge τ=0.5 rows in Table 7. The Pred./Meas. Density columns validate the threshold calibration: measured densities track predictions (Eq. 3) with standard deviation ≤ 0.06 across all configurations and lengths.

Table 6: Full RULER results across multiple sparsity levels. Accuracy column shows absolute accuracy with relative degradation from baseline in brackets. Pred./Meas. Density shows predicted (Eq. 3) vs. measured density during inference (predicted/measured$\pm$std). Speedup is end-to-end TTFT vs. dense FlashAttention-2 on the same samples.

Seq Length	Method	Sparsity Param.	Accuracy ↑	Pred./Meas. Density	Speedup ↑
32K	Dense FlashAttention-2	All	85.98%	1.00	1.00×
	BSFA	k=64	84.94% (-1.2%)	0.24/0.28±0.04	1.07×
		k=96	85.52% (-0.5%)	0.35/0.38±0.05	1.04×
		k=128	86.00% (+0.0%)	0.45/0.47±0.05	1.03×
		k=192	86.42% (+0.5%)	0.62/0.62±0.05	1.00×
	SpargeAttention	τ=0.6	75.62% (-12.1%)	-	0.91×
		τ=0.75	85.83% (-0.2%)	-	0.88×
		τ=0.85	86.18% (+0.2%)	-	0.87×
64K	Dense FlashAttention-2	All	84.96%	1.00	1.00×
	BSFA	k=192	83.08% (-2.2%)	0.35/0.36±0.05	1.13×
		k=256	83.39% (-1.8%)	0.44/0.45±0.05	1.09×
		k=384	84.24% (-0.8%)	0.62/0.61±0.05	1.05×
	SpargeAttention	τ=0.3	66.04% (-22.3%)	-	1.19×
		τ=0.4	75.07% (-11.6%)	-	1.10×
		τ=0.5	83.03% (-2.3%)	-	1.03×
128K	Dense FlashAttention-2	All	76.79%	1.00	1.00×
	BSFA	k=512	73.71% (-4.0%)	0.44/0.44±0.06	1.16×
		k=768	75.75% (-1.4%)	0.61/0.61±0.05	1.11×
		k=1024	76.54% (-0.3%)	0.76/0.74±0.05	1.06×
	SpargeAttention	τ=0.5	70.68% (-8.0%)	-	1.17×
		τ=0.6	75.28% (-2.0%)	-	1.09×
		τ=0.7	75.42% (-1.8%)	-	1.05×

A.2 RULER 64K Speedup, All Methods

Table 7 reports end-to-end TTFT speedup at the canonical RULER 64K cross-section across all methods. RULER accuracy and density are reported in Appendix Table 6 for BSFA and SpargeAttention. The new sparse-attention baselines (MInference, FlexPrefill, BLASST) were not evaluated for RULER accuracy in our experiments.

A.3 Extra Hardware Setup Details

A6000. The A6000 (GA102, SM 8.6) provides $\sim$ 768 GB/s of memory bandwidth versus $\sim$ 2,039 GB/s on the A100 (GA100, SM 8.0), with FP16 tensor-core throughput also lower (76 vs. 312 TFLOPS). To run on SM 8.6, the kernel was rebuilt with `FLASH_ATTN_CUDA_ARCHS="8.6"`. Crucially, we reuse the threshold tensor calibrated on A100 with no recalibration. Speedups in main Table 4 are measured on RULER 64K with 50 samples.

Table 7: End-to-end TTFT speedup on RULER 64K (Llama-3.1-8B). All methods on A100 unless marked $\star$ (H100). XAttention was not measured on RULER.

Method	Param.	Speedup ↑
Dense FA-2	All	1.00×
BSFA	k=192	1.13×
	k=256	1.09×
	k=384	1.05×
SpargeAttention	τ=0.5	1.03×
MInference	default	1.15×
FlexPrefill	γ=0.9	1.70×
	γ=0.99	1.14×
BLASST*	50%	1.14×
	70%	1.17×

H100. We port BSFA to H100 (Hopper, SM 90) using block dimensions $B_M = 128$, $B_N = 224$ to match Hopper's memory hierarchy, with the threshold tensor recalibrated for the new block size. We have implemented BSFA on H100 on top of the same attention algorithm used by FA2. BSFA's porting into FA-3 is left to future work. Speedups in main Table 4 are measured on LongBench longest-10 narrativeqa samples ($\sim$65K tokens each).

A.4 Qwen2.5-7B Details

We evaluate on Qwen2.5-7B-Instruct [Team, 2025] with 28 layers, 28 attention heads, 4 KV heads (GQA-7), and head dimension 128. The threshold tensor stores three sparsity levels ($k \in \{16, 32, 64\}$) and totals 2.3 MB. Main Table 5 reports the resulting LongBench accuracy and RULER 64K speedup.

A.5 Baseline Implementation Details

We document method-specific implementation choices for the sparse-attention baselines.

SpargeAttention. We use the `spas_sage_attn_meansim_topk_cuda` A100 kernel, which combines sparsity with INT8 Q/K quantization. The quantization is orthogonal to our FP16 selective computation. We reached out to the authors to verify our results but received no response.

XAttention. We use the hardcoded LLaMA-3.1-8B thresholds in their `eval/LongBench/ratio.py`. The DP optimizer described in their §2.3 is not in the released code (their GitHub issue #19).

FlexPrefill. FlexPrefill requires bf16, so its TTFT is reported against a dtype-matched bf16 dense baseline.

B Background Details

B.1 Scaled Dot-Product Attention

Transformers use multi-head scaled dot-product attention to process sequences of tokens [Vaswani et al., 2017]. Given a sequence of N tokens with model dimension d_{model}, the input $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{model}}}$ is projected into H attention heads, each with dimension $d = d_{\text{model}}/H$. For each head h:

$$\mathbf{Q}^{(h)} = \mathbf{X}\mathbf{W}_Q^{(h)} \in \mathbb{R}^{N \times d} \quad \text{(queries)} \tag{4}$$

$$\mathbf{K}^{(h)} = \mathbf{X}\mathbf{W}_K^{(h)} \in \mathbb{R}^{N \times d} \quad \text{(keys)} \tag{5}$$

$$\mathbf{V}^{(h)} = \mathbf{X}\mathbf{W}_V^{(h)} \in \mathbb{R}^{N \times d} \quad \text{(values)} \tag{6}$$

where $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)} \in \mathbb{R}^{d_{\text{model}} \times d}$ are learned projection matrices. The attention output for each head is:

$$\text{Attention}^{(h)}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)}) = \underbrace{\text{softmax}\left(\frac{\mathbf{Q}^{(h)}(\mathbf{K}^{(h)})^\top}{\sqrt{d}}\right)}_{\mathbf{P}^{(h)} \in \mathbb{R}^{N \times N}} \mathbf{V}^{(h)} \tag{7}$$

The outputs from all heads are concatenated (yielding dimension $d_{\text{model}} = H \cdot d$) and then passed through a final linear projection $\mathbf{W}_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ to mix information across heads. For simplicity of notation, we omit positional encodings (e.g., RoPE) and focus on a single head in subsequent discussions, dropping the superscript (h).

In standard implementations, the projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$ are applied to the full d_{model}-dimensional input before splitting into heads. The computational costs are:

- **Linear projections**: $O(Nd_{\text{model}}^2)$ FLOPs total for all $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ projections across all heads.
- **Score computation (QK)**: $O(N^2 d)$ FLOPs per head, $O(N^2 d_{\text{model}})$ total.
- **Softmax normalization**: $O(N^2)$ operations per head, $O(N^2 H)$ total.
- **Value aggregation (PV)**: $O(N^2 d)$ FLOPs per head, $O(N^2 d_{\text{model}})$ total.
- **Output projection**: $O(Nd_{\text{model}}^2)$ FLOPs.

For long sequences where $N \gg d_{\text{model}}$, the operations quadratic in N dominate: both the QK score computation and PV aggregation scale as $O(N^2 d_{\text{model}})$ while the linear projections scale as $O(Nd_{\text{model}}^2)$. For example, in Llama-3.1-8B with $d_{\text{model}} = 4096$ ($d = 128$, $H = 32$), processing a sequence of $N = 128$K tokens requires $N^2 d_{\text{model}} \approx 6.7 \times 10^{13}$ operations for each of QK and PV, while the linear projections require only $Nd_{\text{model}}^2 \approx 2.1 \times 10^{12}$ operations, yielding a ratio of approximately 32:1. Therefore, optimizing attention for long contexts primarily requires addressing the $O(N^2 d)$ complexity of the QK and PV operations.

Beyond computational complexity, memory becomes an equally critical bottleneck. Materializing the full $N \times N$ attention matrix $\mathbf{P}$ requires $O(N^2)$ memory, which becomes prohibitive for long sequences. For instance, storing the attention scores for our 128K example would require over 16 billion elements per head, consuming 32GB in FP16 precision for a single head alone.

B.2 FlashAttention-2: Tiled Computation

FlashAttention-2 [Dao, 2023] computes exact attention without materializing the full $N \times N$ score matrix by using a tiled algorithm with online softmax. The key insight is to partition the computation into blocks and maintain running statistics that allow incremental updates. The algorithm leverages the GPU memory hierarchy, keeping intermediate results in fast on-chip SRAM (Static RAM) while streaming data from slower but larger HBM (High Bandwidth Memory).

The algorithm partitions the query sequence into $M_Q = \lceil N/B_M \rceil$ blocks of size B_M and the key/value sequences into $M_{KV} = \lceil N/B_N \rceil$ blocks of size B_N. Each query block $\mathbf{Q}_i \in \mathbb{R}^{B_M \times d}$ (for $i = 0, \ldots, M_Q - 1$) processes all key blocks $\mathbf{K}_j \in \mathbb{R}^{B_N \times d}$ and corresponding value blocks $\mathbf{V}_j \in \mathbb{R}^{B_N \times d}$ (for $j = 0, \ldots, M_{KV} - 1$) in sequence.

The key algorithmic innovation is the use of online softmax with incremental updates. Instead of computing and storing the full attention matrix, FlashAttention processes one block at a time and maintains running statistics (maximum values and normalizers) that are updated as each new block is processed. Conceptually, for each query block $\mathbf{Q}_i$, the algorithm: (1) loads $\mathbf{Q}_i$ from HBM to on-chip memory (shared SRAM and registers) where it remains throughout processing, (2) streams key blocks $\mathbf{K}_j$ from HBM one at a time, (3) computes attention scores $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top / \sqrt{d}$ using tensor cores, with score tiles kept on-chip, (4) updates running statistics (maximum values and normalizers) which are also maintained on-chip, (5) loads the corresponding value block $\mathbf{V}_j$ and accumulates the weighted output in on-chip memory, and (6) rescales the accumulated results to maintain numerical precision. Throughout the inner loop over key/value blocks, all intermediate results stay on-chip. Only after processing all blocks for a given query block $\mathbf{Q}_i$ is the final output

$\mathbf{O}_i$ written back to HBM. The score tiles $\mathbf{S}_{ij}$ are never written to HBM but exist only in on-chip memory during computation. This tiled approach means the full $N \times N$ attention scores matrix is never materialized in memory, fundamentally changing the memory requirements from quadratic to linear. The complete algorithmic details can be found in Dao [2023].

The per-tile computational cost reveals why attention remains expensive even with FlashAttention's memory optimizations. For each tile (i, j), the algorithm performs two equally expensive matrix multiplications: computing the attention scores $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top$ and applying them to values $\tilde{\mathbf{P}}_{ij} \mathbf{V}_j$, each requiring $2B_M B_N d$ FLOPs. The algorithm must also load both $\mathbf{K}_j$ and $\mathbf{V}_j$ blocks from HBM, with each block containing $B_N d$ elements. Additional overhead from computing exponentials and maintaining running statistics is comparatively minor ($O(B_M B_N)$ operations). Since FlashAttention processes $M_Q \times M_{KV}$ total tiles (approximately half for causal attention), and the QK and PV operations equally dominate the computational cost, each contributes roughly $2N^2 d$ FLOPs to the total complexity.

B.3 Calibration and Practical Details

Threshold tensor. We calibrate a tensor $\mathbf{T} \in \mathbb{R}^{S \times L \times H \times \lceil N_{\max}/B_M \rceil}$ where S is the number of sparsity levels, L is the number of layers, H is the number of heads, and $N_{\max}$ is the maximum calibration sequence length. At inference, selecting sparsity level k extracts the slice $\mathbf{T}[k, :, :, :]$, which provides per-layer, per-head, per-position thresholds.

Storage overhead. Storage is minimal: for a single sparsity level and $N_{\max} = 65{,}536$, Llama-3.1-8B requires approximately 5.2×10^5 threshold values, negligible compared to the model's 8B parameters (0.007%).

Practical considerations. For sequences not aligned with block boundaries, we pad by repeating the last token to avoid introducing artificial scores that would distort the top-k selection. Sequences exceeding $N_{\max}$ reuse thresholds from position $N_{\max}$, as attention patterns empirically stabilize in later positions. While presented for causal attention, BSFA extends naturally to bidirectional settings.

Deployment flexibility. Storing multiple sparsity levels enables dynamic adjustment of the latency-accuracy trade-off at deployment time without recalibration. Different applications may prioritize speed versus accuracy differently, and specific tasks like information retrieval can use much higher sparsity while maintaining accuracy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state our main claims (training-free, drop-in replacement for FlashAttention; $\sim$50% compute/bandwidth savings on skipped blocks; up to $1.13\times$ end-to-end speedup on LongBench at 1.1% accuracy drop on Llama-3.1-8B, $1.24\times$ on Needle-in-a-Haystack at 99% accuracy, attention-kernel speedups up to $1.38\times$ at higher sparsity, and end-to-end speedup that stays within a few percent of dense across the full range of practical sequence lengths). The paper additionally reports head-to-head comparison against five recent sparse-attention baselines (SpargeAttention, MInference, FlexPrefill, XAttention, and BLASST) and validation on Qwen2.5-7B and on A6000 and H100 GPUs. These claims are supported by the experiments in Section 4.3 (complexity analysis), Section 5 (Experimental Evaluation), Tables 1–5, and Figures 2–3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the scope and limitations of the approach in the Limitations and Conclusion paragraph at the end of Section 5. BSFA targets long-context prefill rather than decode, and the implementation extends FlashAttention-2 with FA-3 integration left to future work. Threshold calibration assumes the per-(layer, head, position) attention-score distribution is approximately stable across inputs (Section 4.2); cross-model, cross-hardware, and cross-dataset generality of the calibrated thresholds is empirically validated on Qwen2.5-7B and on A6000 and H100 GPUs (Section 5.2, Appendices A.3 and A.4).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain formal theorems; the contribution is an algorithm with empirical evaluation. The complexity argument in Section 4.3 is an operation count and is described in full.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The CUDA kernel, calibration scripts, and evaluation harness are released at the GitHub link in the abstract. Section 5.1 fully specifies the models (Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct), benchmarks (RULER at 32K/64K/128K, full LongBench plus the longest-10 and mixed-160 TTFT subsets, Needle-in-a-Haystack at 64K), GPUs (A100 80GB and A6000 with CUDA 12.1, plus a Hopper-compatible port on H100), block sizes (B_M=128, B_N=64 on A100/A6000 and B_N=224 on H100), precision (FP16), calibration sample counts (16 for Llama-3.1-8B, 21 at 32K for Qwen2.5-7B), measurement protocol (10 runs after warmup on disjoint task categories), and baseline configurations (SpargeAttention τ values, MInference default, FlexPrefill γ values, XAttention default, BLASST sparsity targets, sliding-window sizes). Algorithm 1 describes the kernel modification, and Appendix B.3 details the calibration tensor and the handling of edge cases. Method-specific baseline notes (e.g., BLASST's H100-only release, FlexPrefill's bf16 dtype constraint, XAttention's hardcoded thresholds) are documented in Appendix A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully

might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The CUDA kernel and reproduction scripts are released at the GitHub repository linked in the abstract and Section 5.1. The benchmarks used (RULER, LongBench, Needle-in-a-Haystack) and the evaluation harness (`lm-evaluation-harness`) are publicly available. Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct are publicly available under their respective model licenses, and the baseline implementations (SpargeAttention, MInference, FlexPrefill, XAttention, BLASST) are obtained from their authors' public releases.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The method is training-free. Section 5.1 specifies the calibration/evaluation split, sample sizes, sequence lengths, sparsity parameters k (the only method hyperparameter), block sizes (B_M=128 throughout, B_N=64 on A100/A6000 and B_N=224 on H100), and baseline hyperparameters (SpargeAttention τ values, MInference default pattern assignments, FlexPrefill γ values, XAttention default, BLASST sparsity targets, sliding-window widths). Calibration details and the per-layer/head/position threshold tensor are described in Section 4.2 and Appendix B.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix Table 6 reports measured-density standard deviations across calibration samples in the form predicted/measured $\pm$ std. TTFT measurements are averages over 10 runs after warmup on the same input, as described in Section 5.1; TTFT exhibits low variance across same-length sequences, which we explicitly note when motivating the sample size. We do not report error bars on accuracy because each accuracy number reflects a deterministic evaluation over a full benchmark split.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section 5.1 specifies NVIDIA A100 80GB GPUs with CUDA 12.1, FP16 precision, and the kernel block dimensions optimized for A100 tensor cores. Section 5.2 and Appendix A.3 describe the additional A6000 (Ampere, $\sim$768 GB/s memory bandwidth) and H100 (Hopper, ported with B_N=224) evaluations. TTFT is the wall-clock prefill time per sequence and is reported in Tables 1–5 and Figures 2–3 as a multiplicative speedup against the dense FlashAttention-2 baseline on the same hardware.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work conforms with the NeurIPS Code of Ethics. The paper is purely an inference-efficiency optimization for an existing, publicly released model; it does not involve human subjects, sensitive data collection, or applications that introduce ethical risks beyond those already present in the underlying language model.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The contribution is a drop-in efficiency improvement for long-context inference. Positive impacts include lower energy consumption per inference and broader access to long-context language models on commodity hardware. Negative impacts are indirect and shared with the underlying language models (e.g., misuse of LLMs); the technique itself does not introduce new misuse pathways beyond those already present in the dense baseline.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a CUDA attention kernel; no model weights, datasets, or generative artifacts are released. The kernel does not modify model behavior beyond a configurable accuracy/speed trade-off and does not introduce misuse risk on its own.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party assets are cited at first use. The models are Llama-3.1-8B-Instruct [Grattafiori et al., 2024] (Llama 3.1 Community License) and Qwen2.5-7B-Instruct [Team, 2025] (Apache 2.0). The framework dependencies are FlashAttention-2 [Dao, 2023] (BSD-3-Clause) and `lm-evaluation-harness` [Gao et al., 2024a] (MIT). The benchmarks are RULER [Hsieh et al., 2024] and LongBench [Bai et al., 2024]. The sparse-attention baselines, used per their public releases and respective open-source licenses, are SpargeAttention [Zhang et al., 2025], MInference [Jiang et al., 2024], FlexPrefill [Lai et al., 2025], XAttention [Xu et al., 2025], and BLASST [Yuan et al., 2025]. The released kernel extends FlashAttention-2 and will be released under a compatible license noted in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper releases a new CUDA kernel (BSFA) extending FlashAttention-2, with reproduction scripts and a README describing build/install steps, calibration, and evaluation. The repository is public and linked in the abstract.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the test subject of the proposed inference kernel, not a component of the research methodology. No LLM was used in a non-standard way to develop the method or to produce the experimental results; any use of LLMs by the authors was limited to writing/editing assistance, which does not require declaration per the NeurIPS LLM policy.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.