# THRIFTATTENTION: SELECTIVE MIXED PRECISION FOR LONG-CONTEXT FP4 ATTENTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Efficient attention algorithms are critical to mitigate the quadratic cost of attention in long-context workloads. Such methods often rely on low-bit quantisation or hard sparsity to accelerate inference. FP4 attention on Blackwell GPUs offers large speedups but sacrifices output quality in long-context workloads, while sparsity methods discard query-key interactions outright which can degrade model quality. We address both failure modes through **ThriftAttention**, a mixed-precision attention mechanism that uses a heuristic to promote only the most important interactions to FP16 while computing the remainder in FP4. ThriftAttention relies on our finding that the output impact of quantisation error is highly non-uniform and increases with the importance of each query-key interaction, concentrating functionally relevant error in a small number of attention blocks that contain the most important tokens. Our experiments across long-context benchmarks and model families show that by computing only $5\%$ of the attention blocks in FP16, ThriftAttention recovers on average $92.3\%$ of the FP4→FP16 performance gap. We demonstrate ThriftAttention's advantage as context length grows, the regime where FP4 attention performs worst.

## 1 INTRODUCTION

Efficient inference is critical for the deployment of large language models (Pope et al., 2023; Wan et al., 2024). Attention (Vaswani et al., 2017; Zhang et al., 2025b) is a key bottleneck in long-context workloads, where its quadratic cost and KV-cache memory traffic dominate execution time (Kwon et al., 2023; Patel et al., 2024). NVIDIA's Blackwell architecture (NVIDIA Corporation, 2024) introduces native FP4 Tensor Cores that have $4\times$ the arithmetic throughput of the equivalent FP16 instructions on Blackwell GPUs (NVIDIA Corporation, 2026) whilst reducing KV-cache memory traffic by a similar amount. Recent works, including SageAttention3 (Zhang et al., 2025c), exploit this hardware to accelerate attention.

However, realising these efficiency gains while preserving output quality remains challenging. FP4 attention introduces quantisation error through reduced numerical precision. Sparsity methods (Tang et al., 2024; Zhang et al., 2025e; 2023) offer an alternative route to efficient inference by identifying important query-key interactions and computing only those interactions. However, sparsity approaches must drop at least $75\%$ of KV blocks to match FP4 latency during the generation phase at inference. Such aggressive sparsity ratios can become a major source of performance degradation in inference-only sparsity methods because the error from omitting blocks entirely is irrecoverable.

In this work we propose **ThriftAttention**, a training-free, mixed-precision attention mechanism that maintains FP16 output quality at sub-byte inference efficiency. ThriftAttention uses a simple two-stage approach. A lightweight heuristic is used to score each query-key block pair $\bar{Q}_i \cdot \bar{K}_j$. The top-$k$ highest-scoring blocks are computed in FP16, while the remaining blocks are computed in FP4, with both sets merged using online softmax.

Across four models and three long-context benchmarks, computing $5\%$ of query-key blocks in FP16 recovers on average $92.3\%$ of the FP4 to FP16 quality gap, rising to $96.0\%$ at a $25\%$ budget. Negative log-likelihood (NLL) analysis shows ThriftAttention recovers more of the FP4 → FP16 performance gap as context length increases, consistent with our error capture analysis which shows quantisation error becomes more concentrated in important blocks as context length scales. Our experiments suggest an advantage of mixed precision over hard sparsity. In our systems analysis



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

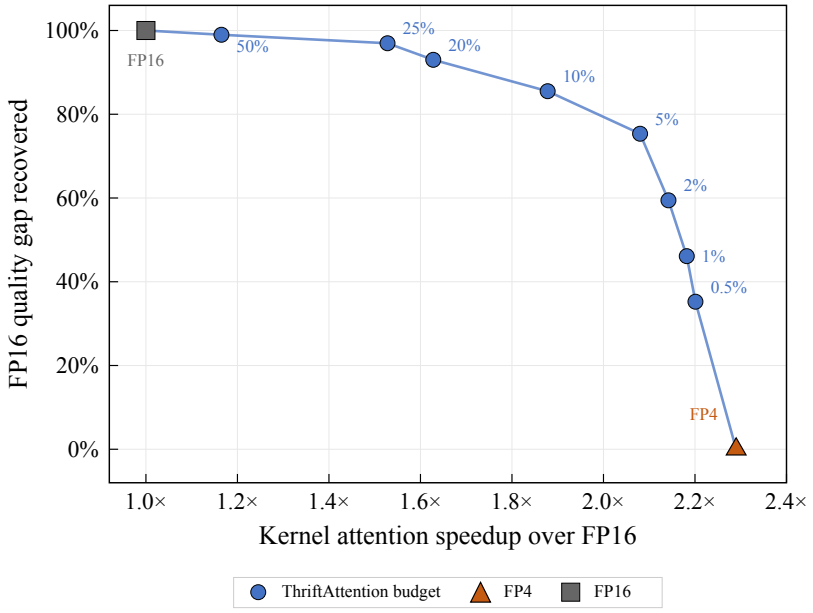



Figure 1: **ThriftAttention approaches FP4 latency while maintaining FP16 quality.** Pareto frontier of NLL recovery versus attention kernel speed-up for Qwen3-8B at 131k context length. Performance recovery is measured as the percentage of the FP4-to-FP16 NLL gap recovered.

section we show that ThriftAttention delivers the lowest time-between-tokens (TBT) of any sparse attention or KV-cache compression method evaluated, while achieving the highest accuracy.

## 2 Related Work

**I/O-Efficient Attention.** FlashAttention (Dao et al., 2022) introduced tiling to reduce GPU memory I/O, with subsequent versions (Dao, 2024; Shah et al., 2024; Zadouri et al., 2026) improving parallelism and adding hardware-specific optimisations.

**Quantised Attention.** While post-training quantisation is well-established for linear layers (Dettmers et al., 2022; Frantar et al., 2022; Lin et al., 2024; Dettmers et al., 2023; Ashkboos et al., 2024; Liu et al., 2025), its extension to attention remains limited. SageAttention (Zhang et al., 2025d;a) accelerates attention via INT8/FP8 quantisation with outlier smoothing, and SageAttention3 (Zhang et al., 2025c) extends this to FP4 on Blackwell using two-level microscaling. Other works target KV cache compression (Liu et al., 2024; Hooper et al., 2024; Lin et al., 2025) or combine quantised matmuls with sparsity (Kang et al., 2024).

**Sparse Attention.** Quest (Tang et al., 2024) performs query-aware KV block selection using coordinate-wise min-max bounds. Token-level eviction and selection strategies (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Jiang et al., 2024) reduce the active KV set, while NSA (Yuan et al., 2025) and SLA (Zhang et al., 2026) learn sparse structure during training. SpargeAttn (Zhang et al., 2025e) combines block-sparsity prediction with quantised attention, skipping near-zero blocks before computing the remainder in INT8/FP8. This is the closest prior work to ours but uses sparsity as the primary acceleration mechanism and does not utilise sub-8-bit numeric formats. Other approaches including linear attention (Katharopoulos et al., 2020; Choromanski et al., 2021; Wang et al., 2020; Qin et al., 2024; Yang et al., 2025b) work less effectively where the attention distribution is peaked.

**Positioning.** ThriftAttention allocates full precision to the most important blocks rather than imposing uniform quantisation. For a given block the error is upper-bounded by FP4 quantisation noise rather than the magnitude of skipped/approximated attention scores as in sparse methods.

# 3 METHOD

## 3.1 MOTIVATION

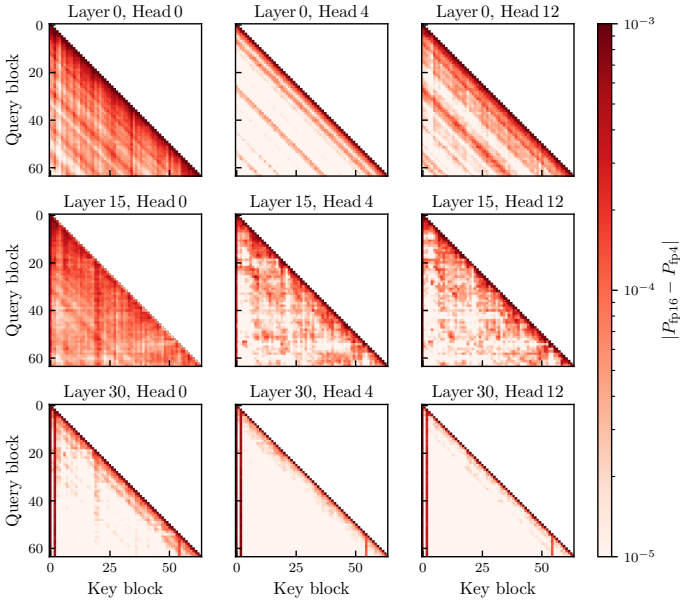



Figure 2: Typical FP16→FP4 attention quantisation error, $e = |P_{\text{FP16}} - P_{\text{FP4}}|$, by query/key blocks across layers and heads in Qwen3-8B.

Consider a single query attending over $N$ keys, with output $o = \sum_{j=1}^{N} p_j v_j$, where $p = \text{softmax}(s)$ and $s_j = q \cdot k_j / \sqrt{d}$. Quantisation perturbs each score to $\tilde{s}_j = s_j + \epsilon_j$. Using the softmax Jacobian, $\partial o / \partial s_j = p_j(v_j - o)$, and defining $\delta o$ as the first-order output perturbation, we have

$$\|\tilde{o} - o\| \approx \|\delta o\| \leq \sum_{j=1}^{N} |\epsilon_j| \, p_j \, \|v_j - o\|. \tag{1}$$

Each key's error contribution is the product of three terms: $|\epsilon_j|$, the score quantisation error; $p_j$, the attention weight; and $\|v_j - o\|$, the value deviation from the output. The $p_j$ factor makes this non-uniform such that tokens with large pre-softmax scores produce large $p_j$ through the softmax exponential, amplifying their own quantisation error. Conversely low attention scores dampen the effect of a key token's quantisation error on $\tilde{o}$. This effect is evident in our analysis of error concentration in Seed-OSS-36B (ByteDance Seed Team, 2025). At a 128k context length, the top 5% of blocks ordered by FP16 attention probability contain 62.1% of the total quantisation error $|P_{\text{FP16}} - P_{\text{FP4}}|$, averaged across layers and heads. Figure 2 shows the non-uniformity of the quantisation error in Qwen3-8B. Error is shown to concentrate in a small number of blocks per query. These are typically near-diagonal blocks and attention sinks, precisely where attention scores are largest.

## 3.2 THRIFTATTENTION METHOD

**Setup.** Let $Q, K, V \in \mathbb{R}^{N \times d}$. Each is quantised to NVFP4 (NVIDIA Corporation, 2024; Rouhani et al., 2023) via $\phi : X \mapsto (X^q, S_X)$, where $X^q$ holds E2M1 elements and $S_X$ holds one E4M3 scale per 16-element group along the contraction axis, so that $S_Q, S_K \in \mathbb{R}^{N \times (d/16)}$ and $S_V \in \mathbb{R}^{(N/16) \times d}$. $\phi_{\text{E2M1}}$ denotes E2M1 rounding without scaling. We partition $Q = [Q_1; \ldots; Q_{T_q}]$ into $T_q = N/B_q$ blocks $Q_i \in \mathbb{R}^{B_q \times d}$, and $K = [K_1; \ldots; K_{T_k}]$, $V = [V_1; \ldots; V_{T_k}]$ into $T_k = N/B_k$ blocks $K_j, V_j \in \mathbb{R}^{B_k \times d}$. We compute the token-wise mean of each block:

$$\bar{Q}_i = \frac{1}{B_q} \sum_{t \in \mathcal{B}_i^Q} Q_t, \qquad \bar{K}_j = \frac{1}{B_k} \sum_{t \in \mathcal{B}_j^K} K_t.$$

**Algorithm 1** ThriftAttention Algorithm

**Input:** $Q, K, V \in \mathbb{R}^{N\times d}$; block sizes $B_q, B_k$; maximum FP16 key blocks per query block $k$

Partition $Q$ into $\{Q_i\}_{i=1}^{T_q}$ and $K, V$ into $\{K_j\}_{j=1}^{T_k}, \{V_j\}_{j=1}^{T_k}$
$(Q^q, S_Q) \leftarrow \mathrm{fp4}(Q)$; $(K^q, S_K) \leftarrow \mathrm{fp4}(K)$; $(V^q, S_V) \leftarrow \mathrm{fp4}(V)$
Compute block means $\{\bar{Q}_i\}_{i=1}^{T_q}$ and $\{\bar{K}_j\}_{j=1}^{T_k}$
**for** $i = 1$ to $T_q$ **do**
  $\mathcal{T}_i \leftarrow \mathrm{TopK}\big(\{\bar{Q}_i \bar{K}_j^\top\}_{j=1}^{T_k}, k\big)$
  *Promoted blocks: FP16 attention*
  $m_i^{16} \leftarrow -\infty$; $\ell_i^{16} \leftarrow \mathbf{0}$; $O_i^{16} \leftarrow \mathbf{0}$
  **for all** $j \in \mathcal{T}_i$ **do**
    $S_{ij} \leftarrow Q_i K_j^\top / \sqrt{d}$
    $m_i^{\mathrm{new}} \leftarrow \max\big(m_i^{16}, \mathrm{rowmax}(S_{ij})\big)$; $\alpha_i \leftarrow \exp\big(m_i^{16} - m_i^{\mathrm{new}}\big)$
    $\widetilde{P}_{ij} \leftarrow \exp\big(S_{ij} - m_i^{\mathrm{new}}\big)$
    $\ell_i^{16} \leftarrow \alpha_i \odot \ell_i^{16} + \mathrm{rowsum}(\widetilde{P}_{ij})$
    $O_i^{16} \leftarrow \mathrm{diag}(\alpha_i) O_i^{16} + \widetilde{P}_{ij} V_j$; $m_i^{16} \leftarrow m_i^{\mathrm{new}}$
  *Remaining blocks: approximate-exponential FP4 path*
  $R_i^4 \leftarrow -\infty$; $\ell_i^4 \leftarrow \mathbf{0}$; $O_i^4 \leftarrow \mathbf{0}$
  **for all** $j \notin \mathcal{T}_i$ **do**
    $S_{ij} \leftarrow \mathrm{Matmul}_{\mathrm{FP4}}\big(Q_i^q, K_j^q, S_{Q,i}, S_{K,j}\big)/\sqrt{d}$
    $x_{ij} \leftarrow \lfloor S_{ij} \log_2 e \rceil$
    $R_i^{\mathrm{new}} \leftarrow \max\big(R_i^4, \mathrm{rowmax}(x_{ij})\big)$; $\beta_i \leftarrow 2^{\,R_i^4 - R_i^{\mathrm{new}}}$
    **for all** 16-element groups $g$ of $x_{ij}$ **do**
      $b_g \leftarrow \mathrm{rowmax}(x_{ij,g})$
      $P_{ij,g}^q \leftarrow \phi_{\mathrm{E2M1}}\big(4 \cdot 2^{\,x_{ij,g} - b_g}\big)$; $S_{P,ij,g} \leftarrow 2^{\,b_g - R_i^{\mathrm{new}}}$
    $\ell_i^4 \leftarrow \beta_i \odot \ell_i^4 + \sum_g S_{P,ij,g} \odot \mathrm{rowsum}(P_{ij,g}^q)$
    $O_i^4 \leftarrow \mathrm{diag}(\beta_i) O_i^4 + \mathrm{Matmul}_{\mathrm{FP4}}\big(P_{ij}^q, V_j^q, S_{P,ij}, S_{V,j}\big)$; $R_i^4 \leftarrow R_i^{\mathrm{new}}$
  *Merge FP16 and FP4 partial states*
  $m_i^\star \leftarrow \max\big(m_i^{16}, R_i^4 \ln 2\big)$
  $a_i^{16} \leftarrow \exp\big(m_i^{16} - m_i^\star\big)$; $a_i^4 \leftarrow \frac{1}{4}\exp\big(R_i^4 \ln 2 - m_i^\star\big)$
  $\ell_i \leftarrow a_i^{16} \odot \ell_i^{16} + a_i^4 \odot \ell_i^4$
  $O_i \leftarrow \mathrm{diag}(\ell_i)^{-1}\big[\mathrm{diag}(a_i^{16}) O_i^{16} + \mathrm{diag}(a_i^4) O_i^4\big]$
**return** $O = [O_1; \ldots; O_{T_q}]$

The importance score for a block pair $(i, j)$ is then

$$\hat{S}_{ij} = \bar{Q}_i \bar{K}_j^\top.$$

**Block routing.** For each query block the $k$ highest-scoring key blocks $\mathcal{T}_i = \mathrm{TopK}_j(\hat{S}_{ij})$ are promoted to FP16:

$$S_{ij} = \begin{cases} Q_i K_j^\top / \sqrt{d}, & j \in \mathcal{T}_i, \\ \mathrm{Matmul}_{\mathrm{FP4}}\big(Q_i^q, K_j^q, S_{Q,i}, S_{K,j}\big)/\sqrt{d}, & j \notin \mathcal{T}_i. \end{cases} \tag{2}$$

Promoted blocks follow the standard FlashAttention-2 online softmax (Dao, 2024) in FP16, returning the per-row maximum $m_i^{16}$, normalisation sum $\ell_i^{16}$, and unnormalised output accumulator $O_i^{16}$. The remaining blocks take the FP4 path outlined below. In the causal path, block selection is restricted to past and current key blocks and the FP16 budget is the percentage of causally valid query-key block pairs computed in FP16.

**FP4 path.** For $j \notin \mathcal{T}_i$, define the base-2 logit $x_{ij} = \lfloor S_{ij} \log_2 e \rceil$. Let $b_g = \max_{r\in g} x_{ij,r}$ be the maximum in the 16-element group $g$ and $R_i$ the running row maximum for that row, both integer-valued by construction. We use an approximate exponential method to build the attention weights

directly in NVFP4:

$$P^q_{ij} = \phi_{\mathrm{E2M1}}\big(4 \cdot 2^{\,x_{ij} - b_g}\big), \qquad s_{P,g} = 2^{\,b_g - R_i}. \tag{3}$$

Both are exact powers of two, so they can be computed on the *ALU* pipe rather than the lower-throughput *MUFU* unit used in the standard exponential operation. Table 1 compares MUFU-based exponential evaluation with our ALU-based weight construction, which achieves over $2\times$ the throughput, with negligible difference in cosine similarity to FP16. The resulting $P^q_{ij}$ and $V^q_j$ are fed directly into the second matmul, along with their microblock scales:

$$O_i \mathrel{+}= \mathrm{Matmul}_{\mathrm{FP4}}\big(P^q_{ij}, V^q_j, s_P, S_{V,j}\big), \tag{4}$$

Since $R_i$ is integer-valued, a row-maximum update rescales $O_i$ by an exact power of two. FP16 and FP4 paths maintain separate partial states and are merged once all blocks have been processed. The non-causal version of the full algorithm is shown in Algorithm 1.

Table 1: Microbenchmarking and accuracy analysis of standard versus our approximate exponential method. Microbenchmark results recorded on an RTX PRO 6000 show that our method achieves $2.16\times$ the throughput of the standard exponential. For cosine similarity, we run ThriftAttention at a 5% FP16 budget with each implementation and compare each output against FP16 attention.

| **Method** | **Ops/Cycle/SM** | **Cosine similarity to FP16** |
|---|---|---|
| Standard exponential (MUFU) | 14.46 | 0.983 |
| Ours (ALU) | 31.23 | 0.982 |

**Implementation and Optimisation on Hardware.** The ThriftAttention mechanism is implemented as a single fused kernel. Within the FP4 path, microblock maxima are reused to compute the online-softmax row maximum, sharing reduction work between quantisation and normalisation. Instead of a row-sum reduction, the normalisation denominator is accumulated through an additional Tensor Core multiplication with an all-ones operand, which is encoded as a register constant and requires no memory loads. The approximate exponential method described above relocates work from the `MUFU` pipe to the higher-throughput `ALU` pipe. Asynchronous value transfers overlap value loading with softmax computation. As in SageAttention3, we permute $K$ during quantisation, so that attention weights can be packed directly into the second matrix multiplication's register layout without needing to redistribute attention weights between threads.

## 4 Experiments

**Experimental Setup.** We evaluate ThriftAttention's accuracy and inference efficiency across multiple model families and long-context benchmarks, comparing against FP16, FP4, and training-free sparse attention methods. We also conduct negative log-likelihood and quantisation error analysis to examine how ThriftAttention's quality recovery and error capture vary with context length.

### 4.1 Inference Efficiency

We evaluate ThriftAttention's inference efficiency through kernel-level and end-to-end systems analysis. For end-to-end evaluation, we use the `mistralai/Ministral-3-8b` model and compare efficiency and downstream quality against SpargeAttention (Zhang et al., 2025e), FlexPrefill (Lai et al., 2025), and MInference (Jiang et al., 2024) during prefill, and QUEST (Tang et al., 2024), RocketKV (Behnam et al., 2025), and SnapKV (Li et al., 2024) during decoding. We evaluate downstream quality on LongBench v2 (Bai et al., 2025).

**Kernel Analysis.** Figure 3 compares ThriftAttention's kernel times to FP16 and SageAttention3 on an RTX PRO 6000. Note for FP16 we use FlashAttention2, the fastest flash-attention variant supported on SM120 architecture. ThriftAttention achieves an average speedup over FP16 of $1.97\times$, $1.83\times$ and $1.51\times$ at $5\%$, $10\%$ and $25\%$ budgets respectively. On average our FP4 kernel achieves a $9.82\%$ lower latency than SageAttention3. This demonstrates the performance benefit of ThriftAttention's exponential approximation, which shifts exponential computation from the low-throughput *MUFU* pipe to the *ALU* pipe on the FP4 path.

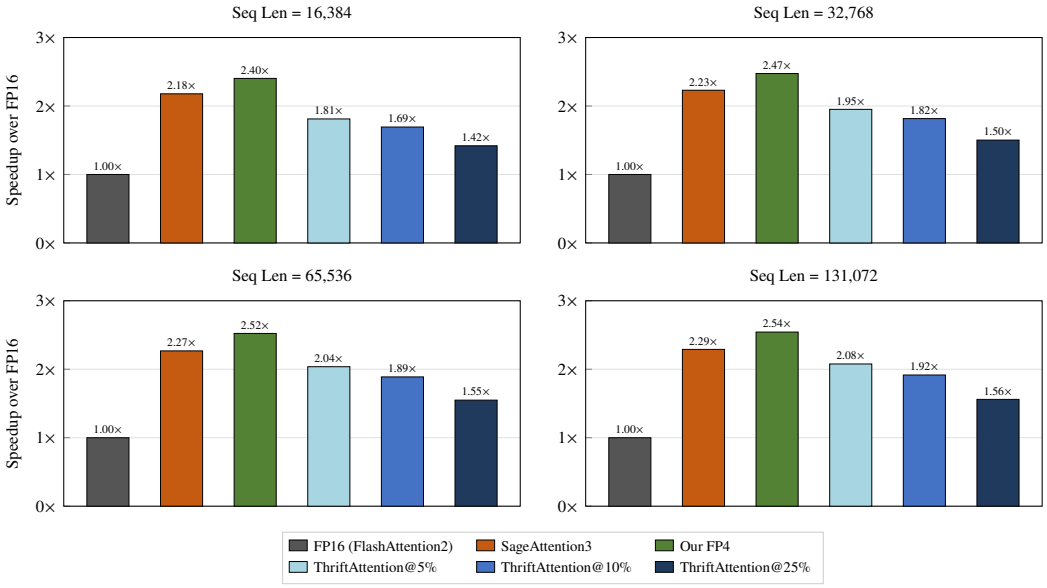



Figure 3: Attention kernel speedup over FP16 (FlashAttention-2) for SageAttention3 and ThriftAttention (5–25% FP16 budgets) on an RTX PRO 6000, with $B = 1$, $n_h = 32$, and $d = 128$.

Table 2: Systems and quality comparison of ThriftAttention prefill performance versus other inference-time attention methods. Results recorded on an RTX Pro 6000.

| Method | Time-to-first-token (s) | Reserved memory (GiB) | Prefill throughput (tokens/s) | LongBench v2 Accuracy (%) |
|---|---|---|---|---|
| FP16 | 4.247 | 21.20 | 7,716.2 | 35.55 |
| FP4 (SageAttention3) | 3.665 | 21.20 | 8,941.0 | 31.89 |
| ThriftAttention @ 5% | 3.685 | 21.65 | 8,892.3 | 34.89 |
| SpargeAttention @25% | 3.706 | 21.32 | 8,841.6 | 29.57 |
| FlexPrefill ($\gamma = 0.95$) | 3.194 | 22.03 | 10,260.8 | 32.23 |
| MInference | 5.384 | 21.20 | 6,086.0 | 34.55 |

**Systems Analysis.** Tables 2 and 3 show the inference efficiency of ThriftAttention in the prefill and decode inference phases. Excluding FP4 and FP16, ThriftAttention achieves the highest accuracy and lowest time-between-tokens (TBT) of the methods tested. For prefill, ThriftAttention has a 13.34% lower throughput than the fastest prefill method, FlexPrefill, however ThriftAttention's output quality is stronger. Table 2 shows that FlexPrefill only marginally outperforms FP4. The

Table 3: Systems and quality comparison of ThriftAttention decoding performance versus other inference-time attention methods. Results recorded on an RTX Pro 6000.

| Method | Time between tokens (ms) | Decode throughput (tokens/s) | Resident memory (GiB) | LongBench v2 Accuracy (%) |
|---|---|---|---|---|
| FP16 | 15.74 | 63.55 | 20.21 | 35.55 |
| FP4 (SageAttention3) | 12.70 | 78.75 | 21.43 | 31.89 |
| ThriftAttention @ 5% | 13.02 | 76.80 | 25.78 | 34.89 |
| QUEST @28.75% | 13.66 | 73.21 | 20.49 | 33.55 |
| RocketKV @1024 | 17.42 | 57.40 | 16.96 | 29.90 |
| SnapKV @1024 | 13.18 | 75.90 | 16.56 | 31.56 |

Table 4: Performance of ThriftAttention at varying FP16 budgets across four models on long-context benchmarks. Recovery measures the fraction of the FP4–FP16 gap closed by each method. For each dataset-model configuration a confidence interval is provided alongside the raw score.

| | | RULER | | LongBench-v1 | | HELMET | |
|---|---|---|---|---|---|---|---|
| Model | Method | Score | Recov. | Score | Recov. | Score | Recov. |
| Llama3.1 (8B) | FP16 | $0.859 \pm 0.006$ | 100.0 | $0.260 \pm 0.007$ | 100.0 | $0.312 \pm 0.008$ | 100.0 |
| | FP4 (SageAttention3) | $0.385 \pm 0.010$ | 0.0 | $0.133 \pm 0.007$ | 0.0 | $0.133 \pm 0.008$ | 0.0 |
| | ThriftAttention @ 5% | $0.770 \pm 0.008$ | 81.2 | $0.271 \pm 0.008$ | 108.7 | $0.287 \pm 0.008$ | 86.0 |
| | ThriftAttention @ 10% | $0.778 \pm 0.008$ | 82.9 | $0.275 \pm 0.008$ | 111.8 | $0.285 \pm 0.008$ | 84.9 |
| | ThriftAttention @ 25% | $0.800 \pm 0.008$ | 87.6 | $0.270 \pm 0.008$ | 107.9 | $0.291 \pm 0.008$ | 88.3 |
| Qwen3 (4B) | FP16 | $0.795 \pm 0.006$ | 100.0 | $0.260 \pm 0.006$ | 100.0 | $0.249 \pm 0.007$ | 100.0 |
| | FP4 (SageAttention3) | $0.445 \pm 0.010$ | 0.0 | $0.134 \pm 0.007$ | 0.0 | $0.130 \pm 0.008$ | 0.0 |
| | ThriftAttention @ 5% | $0.749 \pm 0.007$ | 86.9 | $0.254 \pm 0.006$ | 95.2 | $0.232 \pm 0.005$ | 85.7 |
| | ThriftAttention @ 10% | $0.756 \pm 0.007$ | 88.9 | $0.256 \pm 0.006$ | 96.8 | $0.237 \pm 0.006$ | 89.9 |
| | ThriftAttention @ 25% | $0.768 \pm 0.006$ | 92.3 | $0.257 \pm 0.006$ | 97.6 | $0.246 \pm 0.006$ | 97.5 |
| Qwen3 (8B) | FP16 | $0.840 \pm 0.006$ | 100.0 | $0.246 \pm 0.006$ | 100.0 | $0.259 \pm 0.007$ | 100.0 |
| | FP4 (SageAttention3) | $0.484 \pm 0.010$ | 0.0 | $0.128 \pm 0.007$ | 0.0 | $0.127 \pm 0.008$ | 0.0 |
| | ThriftAttention @ 5% | $0.806 \pm 0.006$ | 90.4 | $0.237 \pm 0.006$ | 92.4 | $0.246 \pm 0.007$ | 90.2 |
| | ThriftAttention @ 10% | $0.815 \pm 0.006$ | 93.0 | $0.241 \pm 0.006$ | 95.8 | $0.252 \pm 0.007$ | 94.7 |
| | ThriftAttention @ 25% | $0.819 \pm 0.006$ | 94.1 | $0.242 \pm 0.006$ | 96.6 | $0.252 \pm 0.007$ | 94.7 |
| Ministral3 (8B) | FP16 | $0.871 \pm 0.005$ | 100.0 | $0.267 \pm 0.011$ | 100.0 | $0.330 \pm 0.009$ | 100.0 |
| | FP4 (SageAttention3) | $0.548 \pm 0.010$ | 0.0 | $0.130 \pm 0.012$ | 0.0 | $0.159 \pm 0.011$ | 0.0 |
| | ThriftAttention @ 5% | $0.866 \pm 0.006$ | 98.5 | $0.258 \pm 0.011$ | 93.4 | $0.329 \pm 0.010$ | 99.4 |
| | ThriftAttention @ 10% | $0.864 \pm 0.006$ | 97.8 | $0.262 \pm 0.011$ | 96.4 | $0.331 \pm 0.010$ | 100.6 |
| | ThriftAttention @ 25% | $0.865 \pm 0.006$ | 98.1 | $0.264 \pm 0.011$ | 97.8 | $0.330 \pm 0.010$ | 100.0 |

main caveat of ThriftAttention is that maintaining separate FP4 and FP16 KV caches incurs a 27.6% memory overhead over FP16 during decoding.

### 4.2 ThriftAttention Benchmark Evaluation

We evaluate ThriftAttention on three benchmarks: LongBench-v1 (Bai et al., 2024), HELMET (Yen et al., 2025), and RULER (Hsieh et al., 2024), across four models (Grattafiori et al., 2024; Yang et al., 2025a; Liu et al., 2026). We benchmark performance against FP4 and FP16, and record the quality gap ThriftAttention recovers between the former and the latter.

**Results.** Table 4 shows the efficacy of ThriftAttention across model families and benchmarks. Promoting only $5\%$ of blocks to FP16 results in an average recovery of $92.3\%$ of the FP4→FP16 performance gap whilst $10\%$ and $25\%$ budgets push this further to $94.5\%$ and $96.0\%$ respectively. Models differ in their required FP16 budgets. Most models show diminishing returns beyond a $5\%$ budget, with the exception of Qwen3-4B, where performance steadily improves as FP16 budget increases.

### 4.3 Sequence Length Experiments

We evaluate ThriftAttention on the HELMET benchmark across sequence lengths $L \in \{8192, \ldots, 131072\}$ and FP16 budgets of 5%, 10% and 25%. We benchmark performance against FP16 and FP4.

**Improvement over FP4.** Table 5 demonstrates that uniform FP4 performs weakly at every length, retaining only 31–58% of the FP16 HELMET score across all models and context lengths. ThriftAttention closes most of this gap across lengths and budgets. At 131k it more than doubles the FP4 score on average, improving over FP4 by 108.6%, 114.4% and 120.2% at 5%, 10% and 25% FP16 budgets. We analyse the sequence length scaling in more detail in our NLL analysis, which is more sensitive to perturbations introduced by quantisation.

Table 5: Per-length HELMET scores of ThriftAttention at varying FP16 budgets across four models.

| Model | Method | 8k | 16k | 32k | 65k | 131k |
|---|---|---|---|---|---|---|
| Llama3.1 (8B) | FP16 | 0.274 | 0.337 | 0.341 | 0.329 | 0.279 |
| | FP4 (SageAttention3) | 0.138 | 0.161 | 0.149 | 0.127 | 0.088 |
| | ThriftAttention @ 5% | 0.265 | 0.331 | 0.312 | 0.310 | 0.216 |
| | ThriftAttention @ 10% | 0.261 | 0.334 | 0.318 | 0.303 | 0.211 |
| | ThriftAttention @ 25% | 0.262 | 0.343 | 0.323 | 0.312 | 0.217 |
| Qwen3 (4B) | FP16 | 0.321 | 0.270 | 0.280 | 0.222 | 0.154 |
| | FP4 (SageAttention3) | 0.175 | 0.156 | 0.162 | 0.084 | 0.073 |
| | ThriftAttention @ 5% | 0.303 | 0.252 | 0.279 | 0.195 | 0.130 |
| | ThriftAttention @ 10% | 0.312 | 0.260 | 0.278 | 0.202 | 0.135 |
| | ThriftAttention @ 25% | 0.318 | 0.268 | 0.275 | 0.222 | 0.148 |
| Qwen3 (8B) | FP16 | 0.277 | 0.301 | 0.272 | 0.232 | 0.214 |
| | FP4 (SageAttention3) | 0.143 | 0.139 | 0.154 | 0.103 | 0.096 |
| | ThriftAttention @ 5% | 0.275 | 0.294 | 0.271 | 0.198 | 0.192 |
| | ThriftAttention @ 10% | 0.268 | 0.298 | 0.271 | 0.216 | 0.209 |
| | ThriftAttention @ 25% | 0.264 | 0.298 | 0.273 | 0.217 | 0.210 |
| Ministral 3 (8B) | FP16 | 0.325 | 0.318 | 0.343 | 0.338 | 0.326 |
| | FP4 (SageAttention3) | 0.141 | 0.155 | 0.169 | 0.174 | 0.157 |
| | ThriftAttention @ 5% | 0.323 | 0.318 | 0.347 | 0.327 | 0.331 |
| | ThriftAttention @ 10% | 0.328 | 0.318 | 0.337 | 0.334 | 0.338 |
| | ThriftAttention @ 25% | 0.323 | 0.318 | 0.344 | 0.331 | 0.334 |

Table 6: NLL recovery and error capture at each context length for Seed-OSS-36B on 300 PG-19 documents. NLL recovery denotes percentage of FP4-FP16 performance gap recovered by each method. Error capture is the proportion of total attention-probability quantisation error stored in the blocks selected by ThriftAttention.

| Method | Metric | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|
| FP4 (SageAttention3) | Error capture (%) | – | – | – | – | – | – |
| | NLL recovery (%) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | ΔNLL vs. FP16 | 0.067 | 0.086 | 0.108 | 0.123 | 0.172 | 0.228 |
| ThriftAttention 5% | Error capture (%) | 38.6 | 39.9 | 44.5 | 49.2 | 53.5 | 57.1 |
| | NLL recovery (%) | 40.3 | 42.6 | 49.9 | 61.2 | 67.4 | 76.6 |
| | ΔNLL vs. FP16 | 0.040 | 0.050 | 0.054 | 0.048 | 0.056 | 0.053 |
| ThriftAttention 10% | Error capture (%) | 45.2 | 52.7 | 55.5 | 59.0 | 63.6 | 66.7 |
| | NLL recovery (%) | 52.6 | 64.4 | 69.1 | 77.3 | 82.8 | 89.3 |
| | ΔNLL vs. FP16 | 0.032 | 0.031 | 0.033 | 0.028 | 0.030 | 0.024 |
| ThriftAttention 25% | Error capture (%) | 66.7 | 69.6 | 72.8 | 75.9 | 79.1 | 81.4 |
| | NLL recovery (%) | 83.3 | 85.5 | 89.4 | 93.8 | 95.8 | 98.0 |
| | ΔNLL vs. FP16 | 0.011 | 0.013 | 0.011 | 0.008 | 0.007 | 0.005 |

### 4.4 Negative Log-Likelihood Analysis

We conduct error capture and negative log-likelihood (NLL) analysis to examine the relationship between quantisation error and recovery and how recovery scales with context length. Error capture is the percentage of total attention-probability quantisation error, $\sum |P_{16} - P_4|$, contained in the blocks selected by ThriftAttention, averaged across heads and layers. We compute error capture and per-token NLL over 300 documents sampled from PG-19 (Rae et al., 2020) across context lengths and FP16 budgets. We use Seed-OSS-36B (ByteDance Seed Team, 2025) for this experiment.

**Advantage of ThriftAttention at longer contexts.** Table 6 reveals a widening advantage for ThriftAttention as context length increases. While the NLL degradation of uniform FP4 relative to FP16 grows by $3.4\times$ from 4K to 128K, ThriftAttention recovers an increasing fraction of this loss at every FP16 budget. Error capture analysis helps explain this trend, showing quantisation

Table 7: Comparison of mixed-precision versus sparse methods on HELMET.

| Type | Method | FP16 (%) | FP4 (%) | Score | Tok/s |
|---|---|---|---|---|---|
| Sparse | Diagonal | 28.75 | 0.0 | 0.101 | 77 |
| Mixed-precision | Diagonal | 5.0 | 95.0 | **0.201** | 79 |
| Sparse | Min-Max (Quest) | 28.75 | 0.0 | 0.181 | 73 |
| Mixed-precision | Min-Max | 5.0 | 95.0 | **0.247** | 73 |
| Sparse | Top-$k$ (SpargeAttn) | 28.75 | 0.0 | 0.169 | 74 |
| Mixed-precision | Top-$k$ | 5.0 | 95.0 | **0.246** | 77 |

error becomes increasingly concentrated in the blocks selected by ThriftAttention as context length grows. At a 25% FP16 budget, error capture rises from 66.7% at 4K to 81.4% at 128K, alongside an increase in NLL recovery from 83.3% to 98.0%. Thus, a fixed relative FP16 budget becomes more effective at longer contexts because the blocks it promotes hold a growing share of the total quantisation error. This relationship is consistent across budgets, where greater error capture corresponds to greater NLL recovery. We hypothesise FP4's degradation reflects an interaction between quantisation error and retrieval difficulty. Long-context prediction depends on fine-grained retrieval against a growing pool of distractor tokens separated by narrower margins, so the perturbation induced by quantisation is increasingly likely to reorder them and blur attention structure.

### 4.5 Comparison between inference-time sparsity methods and mixed-precision at equivalent computational costs

We evaluate mixed-precision methods against inference-time sparsity methods at equivalent compute budgets on the HELMET dataset using Qwen3-8B. We compare mixed-precision approaches using an FP16 budget of $5\%$ to hard sparsity methods with a sparsity ratio of $71.25\%$. Under the $4\times$ FP4-to-FP16 throughput assumption, this gives the same idealised tensor-core cost between sparsity and mixed-precision approaches. For the experiment, the heuristics we use are Diagonal, Quest (Tang et al., 2024) min-max, and SpargeAttn-style (Zhang et al., 2025e) top-$k$.

Table 7 indicates that, at equivalent computation cost, preserving full attention support in FP4 is more effective than hard-sparsity. Sparse methods incur large tail errors since non-selected blocks are removed entirely whereas mixed-precision degrades more smoothly by retaining non-selected blocks in low precision. Thus non-selected blocks are degraded by quantisation rather than deleted. The min-max heuristic slightly outperforms top-k, the default choice of heuristic in ThriftAttention, but we prefer top-k for its $\sim 5\%$ higher throughput.

## 5 Limitations and Future Work

ThriftAttention kernels are designed and optimised for consumer Blackwell GPUs. Extending ThriftAttention to support other hardware would increase the utility of the approach. A natural target is data-centre Blackwell, but experiments included in Appendix B show computing attention in FP4 provides limited throughput benefits on this hardware, since its tensor cores are substantially faster than its other functional units. This makes FP4 attention bottlenecked by softmax computation on data-centre Blackwell, meaning ThriftAttention cannot achieve the same efficiency benefits on this hardware. Integrating ThriftAttention into inference engines is an exciting direction which would allow practitioners to use our approach in production environments.

## 6 Conclusion

We introduce **ThriftAttention**, a mixed-precision attention mechanism for efficient long-context inference which preserves FP16 quality while retaining sub-byte inference efficiency. Our NLL analysis shows this advantage increases with context length as a fixed FP16 budget captures an increasing share of quantisation error. Our experiments suggest retaining non-selected blocks in FP4 can be preferable to deleting them entirely, supporting mixed-precision as a strong alternative for efficient inference.

## REPRODUCIBILITY STATEMENT

The full ThriftAttention algorithm is given in Algorithm 1. Source code for the ThriftAttention kernel and all experiments reported in the paper are provided as supplementary material. Appendix A lists the hardware and software environment for all experiments. Appendix A also provides details of the libraries and frameworks used to reproduce our kernel and systems analysis results. Similarly we provide information on model checkpoints, context and decoding settings, the FP16 budget-to-$k$ mapping and the exact task subsets used for LongBench, RULER, HELMET and the PG-19 NLL experiments. This provides the necessary details to reproduce the results from our benchmark evaluation, sequence level ablation and sparsity versus mixed-precision experiments.

## AI USAGE STATEMENT

In this paper, generative AI tools were used to check for any grammatical errors and assist minor rephrasing. Generative AI was not used to generate any ideas, insights or any other novel contributions provided by the paper. AI was not used to design, implement or test any of the systems contributions of the paper, notably the ThriftAttention kernel. All experiments reported on the paper were designed and implemented by us. We take full responsibility for the final contents of this work.

## REFERENCES


Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, 2024.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137. Association for Computational Linguistics, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. RocketKV: Accelerating long-context LLM inference via two-stage KV cache compression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 3358–3392. PMLR, 2025. URL https://proceedings.mlr.press/v267/behnam25a.html.

ByteDance Seed Team. Seed-oss open-source models. https://github.com/ByteDance-Seed/seed-oss, 2025.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

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. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, 2024.

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? *arXiv preprint arXiv:2404.06654*, 2024.

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. In *Advances in Neural Information Processing Systems*, 2024.

Hao Kang, Srikant Bharadwaj, James Hensman, Tushar Krishna, Victor Ruhle, and Saravan Rajmohan. TurboAttention: Efficient attention approximation for high throughputs LLMs. *arXiv preprint arXiv:2412.08585*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023. doi: 10.1145/3600006.3613165.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OfjIlbelrT.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems*, 2025.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026. doi: 10.48550/arXiv.2601.08584. URL https://arxiv.org/abs/2601.08584.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32332–32344. PMLR, 2024.

NVIDIA Corporation. NVIDIA blackwell architecture technical overview, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture.

NVIDIA Corporation. *Parallel Thread Execution ISA, Version 9.2*, 2026. URL https://docs.nvidia.com/cuda/parallel-thread-execution/. CUDA Toolkit 13.2.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132, 2024. doi: 10.1109/ISCA59077.2024.00019.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024.

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

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, 2024.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey. *Transactions on Machine Learning Research*, 2024.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context language models effectively and thoroughly. In *International Conference on Learning Representations*, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1126.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*, 2026.

Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. SageAttention2: Efficient attention with thorough outlier smoothing and per-thread INT4 quantization. In *International Conference on Machine Learning*, 2025a.

Jintao Zhang, Rundong Su, Chunyu Liu, Jia Wei, Ziteng Wang, Pengle Zhang, Haoxu Wang, Huiqiang Jiang, Haofeng Huang, Chendong Xiang, et al. Efficient attention methods: Hardware-efficient, sparse, compact, and linear attention, 2025b. URL `https://attention-survey.github.io`. Preprint.

Jintao Zhang, Jia Wei, Haoxu Wang, Pengle Zhang, Xiaoming Xu, Haofeng Huang, Kai Jiang, Jianfei Chen, and Jun Zhu. SageAttention3: Microscaling FP4 attention for inference and an exploration of 8-bit training. In *Advances in Neural Information Processing Systems*, 2025c.

Jintao Zhang, Jia Wei, Pengle Zhang, Jun Zhu, and Jianfei Chen. SageAttention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, 2025d.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 76397–76413. PMLR, 2025e.

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jianfei Chen, and Jun Zhu. SLA: Beyond sparsity in diffusion transformers via fine-tunable sparse–linear attention. In *International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=eD8IPvNoZB`.

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. In *Advances in Neural Information Processing Systems*, 2023.

# A EXPERIMENT DESIGN

**Code and environment.** Experiments use CUDA 12.8, PyTorch 2.8.0, and a single NVIDIA RTX PRO 6000 Blackwell GPU with 96GB GPU memory. The full set of reported downstream benchmark evaluations required approximately 600 GPU-hours. The NLL analysis and error capture analysis required approximately 48 GPU-hours. We did not use a larger internal cluster or additional undisclosed compute for the reported results.

**Models.**

| Paper label | Checkpoint ID |
|---|---|
| Llama3.1 (8B) | `meta-llama/Llama-3.1-8B` |
| Qwen3 (4B) | `Qwen/Qwen3-4B` |
| Qwen3 (8B) | `Qwen/Qwen3-8B` |
| Ministral 3 (8B) | `mistralai/Ministral-3-8B-Base-2512` |

**ThriftAttention settings.** All experiments use block sizes $B_q = B_k = 64$. The target FP16 budget fraction is $f \in \{0.05, 0.10, 0.25\}$. For a sequence with $n = \lfloor L/64 \rfloor$ key blocks, the integer top-$k$ is chosen by

$$(kn - k(k-1)/2)/(n(n+1)/2) \approx f,$$

rounded and clamped to $[1, n]$.

**Microbenchmarking, kernels and systems analysis** We compare *MUFU* and *ALU* exponential throughput using standalone CUDA kernels, timing 10,000 iterations with GPU cycle counters and estimating evaluations per cycle per SM from median timings. For our kernel experiments we use 100 warm-up iterations and 1000 timed repetitions with quantisation and block selection included in our kernel timings. Prefill uses upstream SpargeAttention, FlexPrefill (Triton 3.4.0), and MInference 0.1.6 (Transformers 4.57.6; compiled `vllm-flash-attn`). Decode uses a locally implemented SnapKV compression with FlashAttention-2, and upstream RocketKV and QUEST kernels. Model weights use FP16, except FlexPrefill, which uses BF16. All prefill and decode methods are evaluated under equivalent conditions. For end-to-end decoding, all methods use CUDA graph replay, with warm-up and graph capture excluded from the reported timings.

**Benchmarks** For our evaluation benchmarks we evaluate four models (Grattafiori et al., 2024; Yang et al., 2025a; Liu et al., 2026) across sequence lengths $L \in \{8192, \ldots, 131072\}$. Qwen3 uses its native RoPE configuration (Su et al., 2024) up to 32,768 tokens and YaRN scaling (Peng et al., 2024) for longer sequences, while Llama and Ministral use their checkpoint RoPE settings. All generation uses greedy decoding. LongBench-v1 is evaluated through lm-evaluation-harness (Gao et al., 2023) on the English subset: `narrativeqa`, `qasper`, `multifieldqa_en`, `hotpotqa`, `2wikimqa`, `musique`, `gov_report`, `qmsum`, `multi_news`, `trec`, `triviaqa`, `samsum`, `passage_count`, `passage_retrieval_en`, `lcc`, and `repobench-p`; the reported score is `_overall.weighted_avg`. RULER uses the local NVIDIA/RULER port with 100 samples per task-length cell over 13 tasks: 8 NIAH variants, `vt`, `cwe`, `fwe`, `qa_1`, and `qa_2`, at lengths $\{4096, 8192, 16384, 32768, 65536, 131072\}$. HELMET uses the local HELMET config snapshot with 50 samples per task-length cell over recall (`json_kv`), RAG (`kilt_nq`, `kilt_triviaqa`, `kilt_hotpotqa`, `kilt_popqa_3`), reranking (`msmarco_rerank_psg`), LongQA (`narrativeqa`, `infbench_qa`, `infbench_choice`), summarization (`infbench_sum`, `multi_lexsum`), and five ICL tasks, at $\{8192, 16384, 32768, 65536, 131072\}$.

**NLL and sequence-length analysis.** NLL uses 300 documents from `emozilla/pg19`, seed 42, with EOS inserted between documents. We evaluate Seed-OSS-36B (ByteDance Seed Team, 2025) at context lengths $\{4096, 8192, 16384, 32768, 65536, 131072\}$ using teacher-forced next-token cross-entropy; no generation is used. The sequence-length analysis uses the same HELMET settings and greedy decoding as above.

## B FP4 ATTENTION ON DATA-CENTRE BLACKWELL

We compare BF16 FlashAttention-4 (Zadouri et al., 2026) with our SM100-specific FP4 attention kernel on NVIDIA B200. Our kernel computes $QK^\top$ in block-scaled NVFP4 and $PV$ in FP8, using E5M2 attention weights and E4M3 values to eliminate per-block $P$ and $V$ scale factors. We use FP8 for $PV$ to relieve the ALU bottleneck in softmax preparation on SM100. Direct E5M2 encoding of $P$ eliminates FP4 conversion and per-block probability scale-factor computation, reducing ALU pressure. This makes FP8 preferable despite its lower peak Tensor Core throughput, since the kernel is bottlenecked by the ALU pipe. Experiments use CUDA 13.0.1 and the BF16 CuTeDSL implementation of FlashAttention-4 (`flash-attn-4[cu13]==4.0.0b29`, CuTeDSL 4.6.3).

Table 8: Attention kernels on NVIDIA B200 ($B = 1$, $n_h = 16$, $d = 128$; non-causal).

| | FA4 | | FP4 | |
|---|---|---|---|---|
| Sequence length | Mean latency (ms) | TFLOP/s | Mean latency (ms) | TFLOP/s |
| 8,192 | 0.524 | 1049.1 | 0.346 | 1587.3 |
| 16,384 | 1.963 | 1120.1 | 1.296 | 1696.5 |
| 32,768 | 7.892 | 1114.6 | 5.050 | 1741.8 |
| 65,536 | 31.451 | 1118.7 | 19.958 | 1762.9 |
| 131,072 | 130.595 | 1077.7 | 79.313 | 1774.5 |