ARGUS: SEEING BEYOND THE KERNEL IN STAGE-AWARE NVFP4 ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

NVFP4 attention leverages high-throughput FP4 arithmetic on NVIDIA Blackwell GPUs, but existing training-free methods largely optimize the main fused kernel and apply similar low-precision execution across inference stages. This kernel-centric focus overlooks two important bottlenecks: preprocessing and cache construction can constitute a substantial fraction of prefill operator latency, whereas autoregressive decoding is primarily constrained by streaming the growing KV cache. We introduce **ARGUS**, a stage-aware NVFP4 attention design that looks beyond kernel-level acceleration and places low precision where each inference stage benefits from it most. This stage-aware design is realized through two complementary components. **Cache-Ready Prefill** replaces input-dependent statistics with **Offline Sparse Centering**, reducing runtime preprocessing while directly materializing a persistent NVFP4 KV cache. **Cache-Native Decoding** directly consumes this compressed cache with BF16 attention, decoupling cache precision from compute precision to reduce memory traffic without unnecessary low-precision transformations. Across reasoning and long-context evaluations on GQA models, ARGUS maintains competitive model quality and achieves up to **4.09**$\times$ prefill and **2.26**$\times$ decode operator-level speedups, yielding up to **2.40**$\times$ end-to-end acceleration for the evaluated 128K-input workload on SGLang.

1 INTRODUCTION

The efficiency of large language model (LLM) inference is increasingly shaped by attention in long-context workloads. During prefill, attention incurs $O(n^2)$ computational complexity and substantial data movement, whereas decode attention is primarily memory-bandwidth bound due to repeatedly streaming the growing KV cache (Ainslie et al., 2023).

For prefill, substantial efforts have targeted the data movement and computational cost of dense attention without changing the model architecture. Flashattention series (Dao et al., 2022; Dao, 2024; Shah et al., 2024; Zadouri et al., 2026) combine IO-aware tiling with online softmax to compute dense attention without materializing the attention matrix and thus achieve significant acceleration. These IO optimizations leave the quadratic arithmetic complexity of dense attention unchanged. Low-precision attention complements them by accelerating matrix multiplications through higher-throughput low-bit Tensor Core operations (Zhang et al., 2025b; 2024; 2025a; 2026a;b; Sharratt, 2026; Hu, 2026; Li et al., 2026; Shah et al., 2024).

Among low-precision formats, NVFP4 is particularly attractive on NVIDIA Blackwell GPUs (NVIDIA, 2024), combining native FP4 Tensor Core acceleration with compact tensor representations. Existing training-free approaches such as SageAttention3 (Zhang et al., 2026a) improve the numerical robustness of NVFP4 attention through QK smoothing and microscaling quantization and ThriftAttention (Sharratt, 2026) introduce selective higher-precision computation. Nevertheless, these methods still largely treat FP4 attention as a drop-in low-precision replacement for the original scaled dot-product attention.

When applied across prefill and decoding, this kernel-centric design faces two operator-level limitations, as illustrated in Figure 1. First, online smoothing and mixed-precision computation requires additional statistics computation and tensor transformations besides the FP4 matrix multiplications, resulting in considerable preprocessing and cache construction latency. Second, the additional quantization and tensor transformations can offset the arithmetic benefits of FP4 and even result in higher

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.

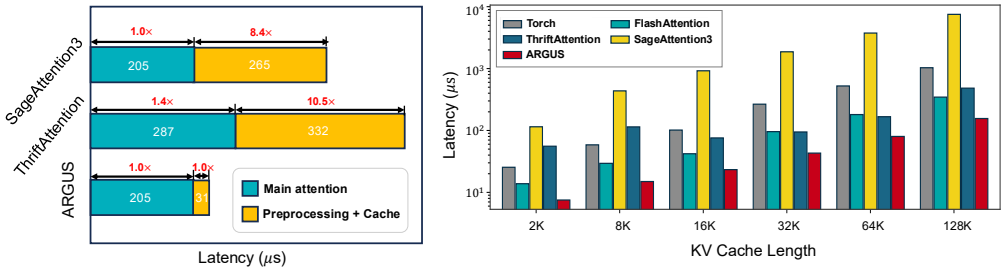


Figure 1: Motivation of ARGUS. **Left:** Operator-level (prefill) breakdown of existing FP4 attention methods. **Right:** Decode operator latency across KV-cache lengths.

latency than BF16 decoding. As a result, directly reusing the prefill-oriented FP4 computation during autoregressive decoding overlooks the bandwidth-bound nature of decoding.

To address these limitations, we propose **ARGUS**, a stage-aware NVFP4 attention design that optimizes LLM inference from an **operator-level** perspective. ARGUS adapts low-precision attention to the distinct characteristics of prefill and decoding, reducing preprocessing overhead for prefilling while exploiting low-precision KV cache for decoding. Our contributions are as follows:

- **Cache-Ready Prefill** with **Offline Sparse Centering**, which selectively retains dominant and stable centering through offline calibration. This design reduces online preprocessing overhead while directly materializing the resulting KV representations into a persistent NVFP4 KV cache.
- **Cache-Native Decoding** is designed for the bandwidth-bound nature of autoregressive generation. Instead of reusing prefill-oriented FP4 computation, ARGUS reads the NVFP4 KV cache and performs BF16 attention, reducing KV cache memory traffic without introducing repeated quantization of previously cached tensors.
- **Extensive experiments:** On the RTX 5090, ARGUS maintains competitive accuracy across the evaluated reasoning and long-context workloads while achieving up to **4.09**× prefill-operator speedup, **2.26**× decode-operator speedup, and **2.40**× end-to-end acceleration for the evaluated 128K-input serving workload on GQA models.

2 Related Work

Efficient Attention. Efficient attention has been studied from both algorithmic and systems perspectives. KV eviction and compression reduce the memory footprint of long-context inference (Zhang et al., 2023; Tang et al., 2024; Li et al., 2024), while sparse attention avoids unnecessary query-key interactions (Jiang et al., 2024; Gao et al., 2024b; Yuan et al., 2025). Architectural designs such as MQA/GQA, MLA, linear attention, and more recent CSA/HCA further reduce KV storage or attention computation (Sun et al., 2023; Team et al., 2025; Xu et al., 2026). Orthogonally, FlashAttention (Dao et al., 2022; Dao, 2024) performs exact dense attention using IO-aware tiling and online softmax (Milakov & Gimelshein, 2018) to minimize HBM traffic, while FlashDecoding and the following work (Dao et al., 2023; Hong et al., 2024) adapts the execution strategy to autoregressive decoding by parallelizing over the KV sequence dimension.

Low-Precision Attention. Low-precision attention follows the FlashAttention paradigm, replacing expensive GEMMs with low-bit Tensor Core operations. FlashAttention-3 (Shah et al., 2024) introduces FP8 attention on Hopper GPUs, the SageAttention series progressively explores lower-precision execution (Zhang et al., 2025b; 2024; 2026a).

FP4 Attention. SageAttention3 extends low-precision attention to Blackwell-native NVFP4, while MXAttention (Yu et al., 2026) explores MXFP4 attention. Hardware-Aware FP4 (Hu, 2026; Zadouri et al., 2026) further adapts FP4 execution to Blackwell hardware. Recent methods improve FP4 accuracy through quantization-aware training (Zhang et al., 2026b), selective higher-precision computation (Sharratt, 2026), or online scale optimization (Gupta et al., 2026). In contrast, ARGUS

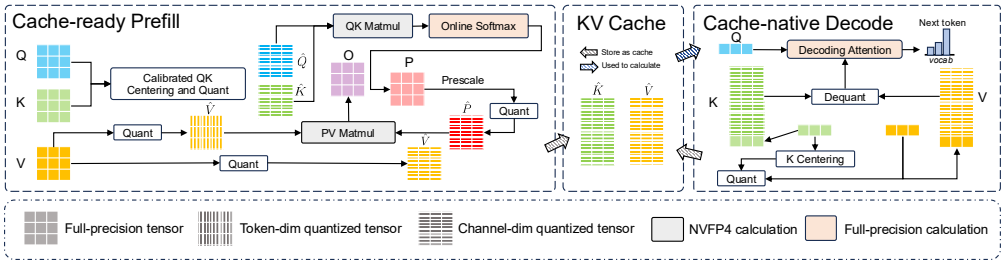


Figure 2: Overview of **ARGUS**. **Cache-Ready Prefill** applies Offline Sparse Centering for NVFP4 attention and directly materializes a persistent NVFP4 KV cache. **Cache-Native Decode** directly consumes the compressed cache and performs BF16 attention computation, decoupling cache precision from compute precision for bandwidth-efficient decoding.

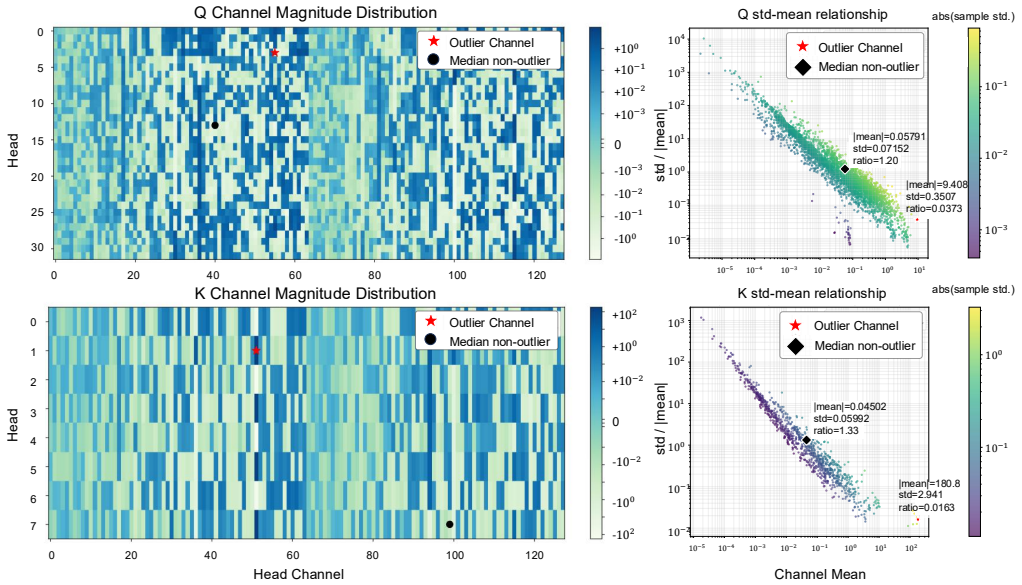


Figure 3: Channel-wise statistics of Q/K activations. **Left:** channel magnitude distributions. **Right:** mean–variation relationships across samples. Dominant channels exhibit larger and more stable offsets, motivating offline-calibrated sparse centering.

focuses on training-free, stage-aware FP4 attention, adapting low-precision execution to compute-intensive prefill and bandwidth-bound decoding. Additional background on FlashAttention and NVFP4 quantization is provided in Appendix A.

3 ARGUS

Existing training-free FP4 attention methods primarily optimize the main attention kernel, assuming that low-precision operands are already available. In practical LLM inference, however, preprocessing and persistent KV-cache construction are also part of the attention operator. We therefore consider operator-level prefill latency as

$$T_{\text{op}}^{\text{prefill}} = T_{\text{prep}} + T_{\text{attn}} + T_{\text{cache}}, \tag{1}$$

where T_{prep} covers statistics collection, centering, quantization, and tensor transformations, and T_{cache} denotes persistent KV-cache materialization.

ARGUS optimizes this complete operator through two stage-specific designs, **Cache-Ready Prefill** and **Cache-Native Decoding**, as illustrated in Figure 2.

Algorithm 1 ARGUS Cache-Ready Prefill

Require: Q, K, V; offline-calibrated sparse centers c_Q, c_K, c_V
Ensure: Attention output O; NVFP4 caches $\mathcal{C}_K, \mathcal{C}_V$

$Q_c \leftarrow Q - \mathbf{1}c_Q^\top, \quad K_c \leftarrow K - \mathbf{1}c_K^\top$ ▷ offline centers, no runtime statistics
$\mathcal{C}_K \leftarrow \mathrm{Quant}^{\mathrm{hd}}_{\mathrm{NVFP4}}(K_c)$ ▷ K representation is directly cache-ready
$\mathcal{C}_V \leftarrow \mathrm{Quant}^{\mathrm{hd16}}_{\mathrm{NVFP4}}(V)$ ▷ append-friendly V cache
Partition Q_c into query tiles $\{Q_i\}$ and (K_c, V) into KV tiles $\{K_j, V_j\}$
for each query tile Q_i **do**
  $\widehat{Q}_i \leftarrow \mathrm{Quant}_{\mathrm{NVFP4}}(Q_i)$
  $m_i \leftarrow -\infty, \quad \ell_i \leftarrow 0, \quad O_i \leftarrow 0$ ▷ online-softmax state
  for each KV tile (K_j, V_j) **do**
    $\widehat{K}_j \leftarrow \mathcal{C}_K[j]$
    $S_{ij} \leftarrow \mathrm{NVFP4MM}(\widehat{Q}_i, \widehat{K}_j^\top) + \mathbf{1}(K_j c_Q)^\top$ ▷ QK + centering compensation
    $m_i' \leftarrow \max(m_i, \mathrm{rowmax}(S_{ij}))$
    $\widetilde{P}_{ij} \leftarrow \exp(S_{ij} - m_i')$
    $\ell_i' \leftarrow e^{m_i - m_i'}\ell_i + \mathrm{rowsum}(\widetilde{P}_{ij})$ ▷ online softmax
    $(\widehat{P}_{ij}, \widehat{V}_j) \leftarrow \mathrm{NVFP4QuantPV}(\widetilde{P}_{ij}, V_j)$ ▷ prescaling + NVFP4 quantization
    $O_i \leftarrow e^{m_i - m_i'}O_i + \mathrm{NVFP4MM}(\widehat{P}_{ij}, \widehat{V}_j)$ ▷ online PV accumulation
    $(m_i, \ell_i) \leftarrow (m_i', \ell_i')$
  end for
  $O_i \leftarrow O_i/\ell_i$
end for
return $O, \mathcal{C}_K, \mathcal{C}_V$

3.1 Cache-Ready Prefill

3.1.1 Offline Sparse Centering

Existing training-free FP4 attention relies on input-dependent Q/K statistics to suppress persistent channel-wise offsets before quantization (Zhang et al., 2026a; Gupta et al., 2026). Computing these statistics online, however, requires sequence-wise reductions and additional tensor passes.

As shown in Figure 3, dominant Q/K channels exhibit large and persistent offsets across inputs. ARGUS therefore estimates static channel-wise means for each layer and attention head from a small calibration set:

$$\mu_X = \frac{1}{|\mathcal{D}_{\mathrm{cal}}|}\sum_{x\in\mathcal{D}_{\mathrm{cal}}}\frac{1}{N_x}X(x)^\top\mathbf{1}_{N_x}, \qquad X \in \{Q, K\}. \tag{2}$$

These statistics are computed once and reused throughout inference.

Let

$$\widetilde{Q} = Q - \mathbf{1}_{N_q}\mu_Q^\top, \qquad \widetilde{K} = K - \mathbf{1}_{N_k}\mu_K^\top. \tag{3}$$

To compensate for the key-dependent term introduced by Q centering, ARGUS computes

$$\begin{aligned}\widetilde{S} &= \frac{\widetilde{Q}\widetilde{K}^\top + \mathbf{1}_{N_q}(\widetilde{K}\mu_Q)^\top}{\sqrt{d}} \\ &= \frac{QK^\top - (Q\mu_K)\mathbf{1}_{N_k}^\top}{\sqrt{d}}.\end{aligned} \tag{4}$$

The remaining term is constant across all keys for each query and therefore cancels under softmax:

$$\mathrm{softmax}(\widetilde{S}) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right). \tag{5}$$

A detailed derivation is given in Appendix B.1.

ARGUS then quantizes the centered Q/K operands to NVFP4 for $QK^\top$, while the subsequent PV computation follows the NVFP4 probability-prescaling and quantization path of SageAttention3.

Figure 3 further shows that large calibrated offsets are substantially more stable across samples than small ones. We therefore retain only offsets that are significant relative to the effective resolution of the underlying FP4 format. For E2M1, the zero-rounding boundary under a scaling target S is

$$\tau_g(S) = \frac{A_g}{4S}. \tag{6}$$

With 4/6 scaling (Cook et al., 2025), $S \in \{4, 6\}$, and hence the finest relative resolution is

$$\min_{S\in\{4,6\}} \frac{\tau_g(S)}{A_g} = \frac{1}{24}. \tag{7}$$

The full derivation is provided in Appendix B.1.1.

ARGUS therefore defines, for each quantization group g,

$$M_g = \max_{j\in g} |\mu_j|, \qquad \tau_g = \frac{M_g}{24}, \qquad c_j = \mu_j \, \mathbb{I}(|\mu_j| \geq \tau_g). \tag{8}$$

Sparse centering retains dominant and stable offsets while leaving small, less stable channels uncentered. Importantly, the equivalence in Eq. 5 holds for arbitrary fixed centering vectors; therefore replacing the dense calibrated means with their sparse versions does not alter the full-precision attention semantics before quantization. All centers and sparsity masks are fixed offline and introduce no additional runtime statistics computation.

3.1.2 Cache-Ready KV Materialization

The low-precision representations used by prefill are not equally suitable for persistent caching. The $QK^\top$ path already quantizes K along the head dimension, so each token is independently represented and the centered NVFP4 K can be directly reused. In contrast, the prefill PV path groups V along the sequence dimension and is therefore incompatible with incremental appends.

ARGUS instead constructs an append-friendly V-cache representation:

$$V_{\text{cache}} = \mathcal{Q}^{\text{hd16}}_{\text{NVFP4}}(V), \tag{9}$$

where head-dimension quantization makes every token independently appendable without requantizing previously cached states.

Algorithm 1 summarizes the resulting Cache-Ready Prefill path, which performs NVFP4 attention while directly materializing the persistent K/V cache for subsequent decoding.

3.2 Cache-Native Decoding

Cache-Native Execution. Autoregressive decoding processes few query tokens while repeatedly streaming the growing KV cache, making memory traffic the dominant bottleneck. ARGUS therefore keeps KV persistently in NVFP4 and dequantizes cache blocks on the load path.

The centered K cache can be consumed directly without reconstructing the original K tensor:

$$K_c = K - \mathbf{1}c_K^\top, \qquad q_t K_c^\top = q_t K^\top - (q_t c_K)\mathbf{1}^\top. \tag{10}$$

The second term is constant across all keys and cancels under softmax. Hence, apart from quantization error, the centered NVFP4 K cache preserves the original attention function.

Stage-Aware Precision Placement. Using an NVFP4 KV cache does not imply that decode computation should also be performed entirely in NVFP4. We decompose decode latency as

$$T_{\text{op}}^{\text{decode}} = T_{\text{load}} + T_{\text{convert}} + T_{\text{compute}}. \tag{11}$$

Since decoding is largely bandwidth-bound, lower compute precision is useful only when its reduction in T_{compute} outweighs the additional conversion overhead. This is particularly challenging for a fully NVFP4 path, because the append-friendly V-cache layout does not directly match the sequence-dimension grouping required by NVFP4 PV, introducing additional regrouping or requantization.

ARGUS therefore decouples KV-cache storage precision from decode compute precision. As shown in Section 4.4, **NVFP4 prefill and KV cache with BF16 decode** provides the best overall accuracy–efficiency trade-off, and is adopted as the default configuration. We further explore hybrid fp4 and fp8 decode variants in the Appendix E.

Algorithm 2 ARGUS Cache-Native Decoding

Require: Decode states q_t, k_t, v_t; persistent NVFP4 caches C_K, C_V; offline sparse center c_K
Ensure: Attention output o_t; updated caches C_K, C_V

for each KV-cache block $(C_K^{(j)}, C_V^{(j)})$ **do**
  $K_j \leftarrow \mathrm{DequantBF16}(C_K^{(j)})$
  $V_j \leftarrow \mathrm{DequantBF16}(C_V^{(j)})$
  $\mathrm{BF16AttentionUpdate}(q_t, K_j, V_j)$
end for
$\mathrm{BF16AttentionUpdate}(q_t, k_t - c_K, v_t)$ ▷ current token
$o_t \leftarrow \mathrm{FinalizeAttention}$
$\hat{k}_t \leftarrow \mathcal{Q}^{\mathrm{hd16}}_{\mathrm{NVFP4}}(k_t - c_K)$
$\hat{v}_t \leftarrow \mathcal{Q}^{\mathrm{hd16}}_{\mathrm{NVFP4}}(v_t)$
$C_K \leftarrow \mathrm{Append}(C_K, \hat{k}_t), C_V \leftarrow \mathrm{Append}(C_V, \hat{v}_t)$ ▷ cache-native update
return o_t, C_K, C_V

Table 1: Accuracy (%) on GSM8K, AIME25, and GPQA-Diamond. Bold indicates the best low-precision result.

Model	Method	GSM8K ↑	AIME25 ↑ 30 × 8 samples	GPQA-Diamond ↑ 198 × 4 samples
Qwen3-4B	BF16	85.37	65.00	53.28
	Sage3	83.70	**65.42**	51.64
	Thrift	**84.00**	63.75	51.01
	ARGUS	83.93	63.33	**51.77**
Qwen3-8B	BF16	89.08	67.75	58.96
	Sage3	85.97	–	–
	Thrift	85.44	67.08	**57.95**
	ARGUS	**88.93**	**68.75**	56.82

4 Experiments

4.1 Experimental Setup

Hardware and Implementation. We conduct the main Grouped Query Attention (GQA) experiments on an NVIDIA RTX 5090 GPU. All performance claims in this section refer to this measured platform unless stated otherwise. ARGUS is implemented in CUDA and CUTLASS (Thakkar et al., 2023). Unless otherwise specified, all latency measurements include the complete attention operator rather than only the fused attention kernel, following the operator boundary defined in Section 3. End-to-end inference experiments are integrated into SGLang (Zheng et al., 2024).

Models and Benchmarks. For the details about models and benchmarks, refer to Appendix D.

Baselines. We compare ARGUS against BF16 attention (FlashAttention2 (Dao, 2024) and SGLang implementation), SageAttention3, and ThriftAttention, representing full-precision, training-free NVFP4, and mixed-precision FP4 attention, respectively. We report both accuracy and operator-level efficiency whenever an executable implementation is available. In addition, the precision-placement study includes a KV-only baseline (BF16 prefill + NVFP4 KV cache + BF16 decode) to isolate the contribution of compressed cache storage from the cross-stage Cache-Ready Prefill path. Detailed generation settings, calibration configurations, and benchmark hyperparameters are provided in Appendix D.

4.2 Effectiveness

We first evaluate whether the operator-level optimizations of ARGUS preserve the quality of the original model. Across both short- and long-context benchmarks, ARGUS consistently remains

Table 2: LongBench results on Llama-3.1-8B-Instruct and Qwen3-8B. Higher is better for all task scores and the average (↑).

	Single-QA		Multi-QA		Sum.		Few-shot			Synthetic		Code		Avg.
Method	Qasp.	MF	HPQ	2W	Gov	MN	TREC	TQA	SAM	PC	PR	LCC	RB-P	
Llama-3.1-8B-Instruct														
BF16	13.30	20.87	20.47	16.53	18.99	7.12	21.67	10.58	4.52	3.33	75.00	10.60	5.02	**17.51**
Sage3	11.79	19.41	21.63	13.98	18.25	8.72	16.67	10.84	4.43	12.33	43.00	10.65	7.10	15.21
Thrift	13.82	20.62	17.70	14.60	19.18	9.17	22.00	11.94	4.09	13.00	20.33	7.76	5.31	13.54
ARGUS	11.71	21.74	15.96	14.93	18.89	8.20	23.67	12.63	4.42	9.67	39.00	10.80	6.63	**15.07**
Qwen3-8B														
BF16	12.77	25.32	12.38	10.42	16.13	13.05	56.00	18.84	5.18	17.33	90.33	11.77	10.83	23.24
Sage3	13.03	25.30	12.46	12.03	16.22	13.23	36.00	18.66	5.00	14.33	80.67	10.92	10.71	20.64
Thrift	12.53	24.13	10.91	10.30	15.92	13.08	51.67	17.68	5.09	14.00	59.33	10.32	10.78	19.65
ARGUS	12.86	25.71	12.66	11.18	16.35	13.10	56.67	18.31	5.18	12.33	87.33	11.32	10.97	**22.70**

Table 3: LongBench-v2 and RULER accuracy (%) on Qwen3-8B and Llama-3.1-8B-Instruct. RULER uses the corresponding RoPE scaling configuration at each context length. Higher is better for all task scores (↑).

	LongBench-v2						RULER					
Method	**Easy**	**Hard**	**Short**	**Med.**	**Long**	**Overall**	**4K**	**8K**	**16K**	**32K**	**64K**	**Avg.**
Qwen3-8B												
BF16	37.50	32.15	40.56	31.16	29.63	34.19	85.02	80.46	81.72	79.37	–	81.64
Sage3	33.33	**28.62**	36.11	26.98	27.78	30.42	84.81	**81.96**	74.58	68.64	–	77.50
Thrift	33.85	**28.62**	34.44	**27.44**	30.56	30.62	84.24	81.77	**80.28**	70.84	–	79.28
ARGUS	**38.02**	27.01	**36.67**	26.05	**32.41**	**31.21**	**85.23**	80.44	80.20	**80.23**	–	**81.52**
Llama-3.1-8B-Instruct												
BF16	29.69	28.62	32.78	27.44	25.93	29.03	90.51	89.59	89.91	84.61	82.83	87.49
Sage3	**32.29**	27.33	31.11	29.30	**25.93**	29.22	88.57	**90.10**	**89.87**	**82.53**	**78.21**	**85.86**
Thrift	27.60	25.72	30.00	25.58	22.22	26.44	84.84	77.34	70.77	59.26	28.70	64.18
ARGUS	28.65	**31.19**	**34.44**	**29.77**	24.07	**30.22**	**89.89**	89.58	88.82	82.13	77.61	85.61

close to the BF16 reference and generally provides a more favorable accuracy profile than existing training-free FP4 attention methods. Importantly, ARGUS does not rely on quantization-aware retraining or online mixed-precision block selection.

Reasoning Tasks. Table 1 reports results on GSM8K, AIME25 and GPQA-Diamond. The results across four model families on GSM8K are reported in Appendix F. ARGUS remains competitive with BF16 and other low-precision baselines across these settings.

Long-Context Tasks. Tables 2 and 3 report results on LongBench-v1/v2. ARGUS consistently remains closer to the BF16 reference than existing FP4 baselines on aggregate long-context accuracy. RULER results in Table 3 further evaluate robustness as the context length increases. In particular, ARGUS avoids the severe long-context degradation exhibited by several existing low-precision methods and maintains stable accuracy across the evaluated context lengths.

4.3 Efficiency

Operator-level efficiency. We evaluate standalone prefill and decode operators. For prefill, ARGUS achieves up to $4.09\times$ speedup over FlashAttention, $2.58\times$ over SageAttention3, and $3.16\times$ over ThriftAttention. For decoding, the corresponding speedups are $2.26\times$, $48.15\times$, and $7.63\times$.

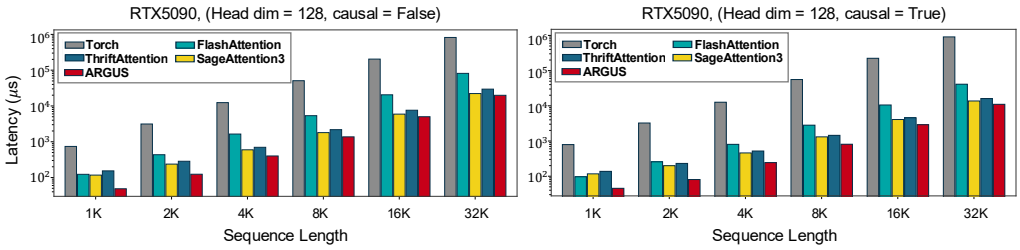


Figure 4: Prefill operator latency across sequence lengths on RTX 5090. **Left:** Non-causal results. **Right:** Causal results.

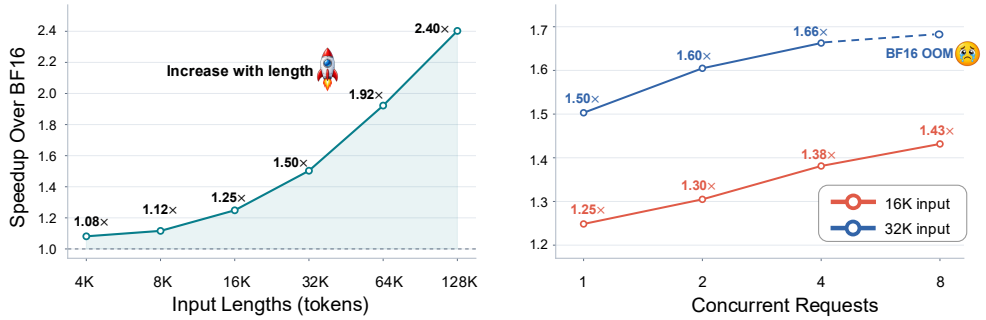


Figure 5: End-to-end inference speedup in SGLang on Qwen3-4B with 128 decode tokens per-request.

The large gain over SageAttention3 mainly comes from the reduced traffic of ARGUS's compressed KV representation, which better matches the KV-cache streaming bottleneck in autoregressive decoding.

End-to-end efficiency. Integrated into SGLang on Qwen3-4B and the RTX 5090, ARGUS achieves up to $2.40\times$ end-to-end speedup over the BF16 baseline at the evaluated 128K input length; the gain is workload-dependent and increases with the KV-cache length. End-to-end latency with the Transformers backend is additionally reported in Table 13.

4.4 Ablation Study

Stage-wise precision placement. Table 4 shows that the benefit of low precision strongly depends on the inference stage. NVFP4 prefill (A1) improves prefill latency but does not accelerate decoding. We treat A2, which uses BF16 prefill with an NVFP4 KV cache and BF16 decode, as a KV-only baseline: it isolates the decode benefit of compressed cache traffic and leaves prefill unchanged. ARGUS (A3) combines this cache representation with the Cache-Ready Prefill path, achieving $3.23\times$ and $2.04\times$ speedups for prefill and decode respectively. In contrast, extending NVFP4 to decode arithmetic (A4) further degrades accuracy and slows decoding to $0.62\times$. These results motivate our final design of NVFP4 prefill and KV-cache storage with BF16 decode computation, which provides the strongest overall accuracy–efficiency trade-off.

Accuracy-preserving techniques. As shown in Table 5, QK centering accounts for the largest reduction in quantization error. Adding 4over6 (Cook et al., 2025) further improves all reconstruction metrics, demonstrating that the two techniques are complementary. Sparse centering retains essentially identical reconstruction accuracy to dense centering with 4over6 in our ablation. We use it as a quantization-aware calibration heuristic to avoid applying corrections to small, less stable offsets. Therefore, sparse QK centering together with 4over6 is adopted in ARGUS.

Table 4: Ablation of precision placement across inference stages. Accuracy metrics measure the deviation from the BF16 reference. Decode speedup is measured at a 4K KV-cache length.

Config.	Prefill	Cache	Decode	L1↓	MSE↓	CosSim↑	Prefill Sp.↑	Decode Sp.↑
BF16	BF16	BF16	BF16	0.0000	0.0000	1.00000	1.00×	1.00×
A1	NVFP4	BF16	BF16	0.2516	0.2150	0.99417	3.14×	1.00×
A2	BF16	NVFP4	BF16	**0.1178**	**0.0872**	**0.99748**	1.00×	**2.04**×
A3 (ARGUS)	**NVFP4**	**NVFP4**	**BF16**	0.3191	0.2846	0.99197	**3.23**×	**2.04**×
A4	NVFP4	NVFP4	NVFP4	0.3543	0.3299	0.99025	**3.23**×	0.62×

Table 5: Ablation study on the accuracy-preserving components of ARGUS.

Config.	L1 ↓	MSE ↓	CosSim ↑
NVFP4 prefill + cache, BF16 decode	0.040130	0.012553	0.970286
+ QK centering	0.021359	0.003569	0.991215
+ QK centering + 4over6	0.019725	0.003051	0.992234
+ Sparse QK centering + 4over6	**0.019652**	**0.003025**	**0.992238**

5 Discussion and Limitations

Favorable workloads. ARGUS is most effective when attention dominates request latency, particularly for long-input prefill and decoding over long KV caches. Short-context requests dominated by projections, MLPs, or framework overhead see smaller end-to-end gains.

Hardware-dependent priorities. Our RTX 5090 experiments demonstrate benefits in both stages. On RTX PRO 6000 Blackwell, stronger BF16 compute at RTX 5090-like bandwidth makes cache-efficient long-context inference a promising target, consistent with ThriftAttention's results (NVIDIA Corporation, 2025b; Sharratt, 2026). For B200/GB200, we expect useful long-prefill gains from an adapted backend, but probability construction and on-chip dataflow must be optimized alongside FP4 matrix operations (Zadouri et al., 2026; Hu, 2026; Li et al., 2026). On GB300, the increased FP4 and exponential throughput makes operand delivery and residual dependencies important optimization targets (Aubrey & Stam, 2025). On Jetson Thor, long KV cache dominated generation can be the clearest opportunity: lower external-memory bandwidth increases the time saved by removing KV traffic, provided load-path dequantization does not consume those savings (NVIDIA Corporation, 2026a). Thus, the deployment priority is probability/dataflow optimization for datacenter prefill and conversion-efficient cache streaming for bandwidth-constrained decoding, rather than uniformly lowering compute precision in both stages.

Scope. Our performance evaluation is restricted to the NVIDIA RTX 5090 due to limited access to other hardware platforms. Appendix H supports the cross-device expectations with hardware specifications, published kernel measurements, and operator-to-request latency accounting while target-device ARGUS validation and generalization to MLA, larger models, and other attention mechanisms remain future work. The end-to-end serving study is centered on Qwen3-4B on the RTX 5090, while the broader evaluation covers additional models and workloads at the accuracy and operator levels.

6 Conclusion

We present ARGUS, a stage-aware NVFP4 attention framework that optimizes low-precision attention at the operator level rather than focusing solely on the main attention kernel. By combining calibrated sparse centering for prefill with a persistent NVFP4 KV cache and BF16 decoding, ARGUS adapts precision placement to the distinct characteristics of prefill and decoding. On the RTX 5090, ARGUS achieves up to $4.09\times$ prefill operator speedup, $2.26\times$ decode operator speedup, and $2.40\times$ end-to-end speedup for the evaluated 128K-input serving workload, while maintaining competitive accuracy with task- and model-dependent degradation.

AI USE STATEMENT

We used generative AI tools to assist with the PyTorch implementation and parts of the CUDA implementation, provide limited feedback on experimental design and settings, and assist with translation. We additionally used these tools to suggest experimental parameters, identify relevant literature, and improve the readability of the manuscript. We did not use generative AI for other required-disclosure tasks applicable to this work.

All AI-assisted code was reviewed and tested by the authors. Suggested experimental settings were independently validated through experiments or checked against original papers and documentation, and literature references were verified against the original sources. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work focuses on efficient LLM inference and does not involve human subjects or the collection of personal or sensitive data. Our experiments use publicly available models and benchmarks under their respective licenses and terms of use.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the ARGUS design and its quantization, calibration, and KV-cache procedures in the main text and appendix. The experimental setup, including model and dataset configurations, hardware settings, evaluation protocols, and additional implementation details, is also documented in the appendix. Together, these details are intended to facilitate reproduction of both the accuracy and efficiency evaluations.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Kyle Aubrey and Nick Stam. Inside NVIDIA Blackwell Ultra: The chip powering the AI factory era. NVIDIA Technical Blog, August 2025. URL https://developer.nvidia.com/blog/inside-nvidia-blackwell-ultra-the-chip-powering-the-ai-factory-era/. Published: August 22, 2025; Accessed: 2026-09-25.

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)*, pp. 3119–3137, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

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. *arXiv preprint arXiv:2412.15204*, 2024b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, Keith Wyss, Mahdi Nazemi, Asit Mishra, Carlo del Mundo, Tijmen Blankevoort, and Song Han. Four over six: More accurate nvfp4 quantization with adaptive block scaling. *arXiv preprint arXiv:2512.02010*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-decoding for long-context inference. https://crfm.stanford.edu/2023/10/12/flashdecoding.html, 2023. Online.

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, 07 2024a. URL https://zenodo.org/records/12608602.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein,

Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tanmaey Gupta, Hayden Prairie, Xiaoxia Wu, Reyna Abhyankar, Qingyang Wu, Austin Silveria, Pragaash Ponnusamy, Jue Wang, Ben Athiwaratkun, Leon Song, et al. Search your block floating point scales! *arXiv preprint arXiv:2605.12464*, 2026.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. FlashDecoding++: Faster Large Language Model Inference on GPUs, January 2024. URL http://arxiv.org/abs/2311.01282. arXiv:2311.01282 [cs.LG].

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.

Robert Hu. Hardware-aware fp4 flashattention-4. *arXiv preprint arXiv:2609.04105*, 2026.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Xingyang Li, Dongyun Zou, Shining Zhang, Jiacheng Chen, Haocheng Xi, Lvmin Zhang, Jun-Yan Zhu, Song Han, Zhekai Zhang, Yujun Lin, et al. Vc-attention: Value smoothing and softmax casting for low-bit attention. *arXiv preprint arXiv:2609.15810*, 2026.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

NVIDIA. Nvidia blackwell architecture technical brief, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture. Technical brief, accessed 2026-04-13.

NVIDIA Corporation. NVIDIA DGX B200: Technical specifications. https://www.nvidia.com/en-us/data-center/dgx-b200/, 2024a. Accessed: 2026-09-25.

NVIDIA Corporation. NVIDIA GB200 NVL72: Technical specifications. https://www.nvidia.com/en-us/data-center/gb200-nvl72/, 2024b. Accessed: 2026-09-25.

NVIDIA Corporation. NVIDIA GB300 NVL72: Technical specifications. https://www.nvidia.com/en-us/data-center/gb300-nvl72/, 2025a. Accessed: 2026-09-25.

NVIDIA Corporation. NVIDIA RTX Blackwell GPU Architecture. White paper, NVIDIA Corporation, 2025b. URL https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf. Version 1.1.

NVIDIA Corporation. NVIDIA RTX PRO Blackwell GPU Architecture. Technical report, NVIDIA Corporation, 2025c. URL https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/NVIDIA-RTX-Blackwell-PRO-GPU-Architecture-v1.0.pdf. Version 1.0. Appendix A, Table 4, pp. 45–46: RTX PRO 6000 Blackwell Workstation Edition specifications.

NVIDIA Corporation. NVIDIA Jetson Thor Series Modules Data Sheet. Technical Report DS-11945-001, Version 1.5, NVIDIA Corporation, 2026a. URL https://developer.download.nvidia.cn/assets/embedded/secure/jetson/thor/docs/Jetson_Thor_Series_Modules_Datasheet_DS-11945-001v1.5.pdf. Accessed: 2026-09-25.

NVIDIA Corporation. *Parallel Thread Execution (PTX) ISA Version 9.4*. NVIDIA Corporation, September 2026b. URL https://docs.nvidia.com/cuda/parallel-thread-execution/index.html. CUDA Toolkit Documentation.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

Joe Sharratt. Thriftattention: Selective mixed precision for long-context fp4 attention. *arXiv preprint arXiv:2605.23081*, 2026.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Aditya Atluri, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, January 2023. URL https://github.com/NVIDIA/cutlass.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025. URL http://arxiv.org/abs/2505.09388. arXiv:2505.09388 [cs].

Jianlin Yu, Jing Lin, Linghui Kong, Aiyue Chen, Weiyi Sun, Chenyu Zeng, Wangli Lan, Jinxi Li, Zhuo Zheng, Ziyang Yue, et al. Mxattention: Data-free optimal scaling and pre-normalization quantization for mxfp4 attention. *arXiv preprint arXiv:2607.24377*, 2026.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. 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)*, pp. 23078–23097, 2025.

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. *arXiv preprint arXiv:2411.10958*, 2024.

Jintao Zhang, Xiaoming Xu, Jia Wei, Haofeng Huang, Pengle Zhang, Chendong Xiang, Jun Zhu, and Jianfei Chen. Sageattention2++: A more efficient implementation of sageattention2. *arXiv preprint arXiv:2505.21136*, 2025a.

Jintao Zhang, Pengle Zhang, Jun Zhu, Jianfei Chen, et al. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, volume 2025, pp. 71566–71585, 2025b.

Jintao Zhang, Haoxu Wang, Pengle Zhang, Xiaoming Xu, Haofeng Huang, Kai Jiang, Jianfei Chen, Jun Zhu, et al. Sageattention3: Microscaling fp4 attention for inference and an exploration of 8-bit training. *Advances in Neural Information Processing Systems*, 38:53901–53931, 2026a.

Peiyuan Zhang, Matthew Noto, Wenxuan Tan, Chengquan Jiang, Will Lin, Wei Zhou, and Hao Zhang. Attn-qat: 4-bit attention with quantization-aware training. *arXiv preprint arXiv:2603.00040*, 2026b.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

A BACKGROUND

A.1 FLASHATTENTION

For completeness, we briefly review the FlashAttention execution model (Dao et al., 2022; Dao, 2024) used throughout this work. We omit batch and head indices for clarity; for GQA, the same formulation applies independently to each query head and its associated KV head.

Given $Q \in \mathbb{R}^{N_q \times d}$, $K, V \in \mathbb{R}^{N_k \times d}$, standard scaled dot-product attention computes

$$S = \frac{QK^\top}{\sqrt{d}}, \qquad P = \mathrm{softmax}(S), \qquad O = PV, \tag{12}$$

where d is the head dimension. For causal attention, the corresponding mask is applied to S before softmax and is omitted here for clarity.

Materializing the full $N_q \times N_k$ score and probability matrices incurs substantial memory traffic. FlashAttention instead partitions Q into query blocks $\{Q_i\}$ and K, V into KV blocks $\{K_j, V_j\}$, and evaluates

$$S_{ij} = \frac{Q_i K_j^\top}{\sqrt{d}} \tag{13}$$

tile by tile while maintaining an online-softmax state.

Specifically, for each query block Q_i, FlashAttention maintains the row-wise running maximum m_i, normalization term ℓ_i, and output accumulator O_i. Given a new score block S_{ij}, the state is updated as

$$m_i' = \max(m_i, \mathrm{rowmax}(S_{ij}))\,, \tag{14}$$

$$\widetilde{P}_{ij} = \exp(S_{ij} - m_i'), \tag{15}$$

$$\ell_i' = e^{m_i - m_i'}\ell_i + \mathrm{rowsum}(\widetilde{P}_{ij}), \tag{16}$$

$$O_i' = e^{m_i - m_i'}O_i + \widetilde{P}_{ij}V_j. \tag{17}$$

After all KV blocks have been processed, the attention output for the query block is obtained as

$$O_i \leftarrow \frac{O_i}{\ell_i}. \tag{18}$$

This tiled online-softmax formulation avoids materializing the full S and P matrices in HBM. Algorithm 1 follows the same execution structure, while replacing the two matrix multiplications with the corresponding low-precision paths.

A.2 NVFP4 QUANTIZATION

NVFP4 is a Blackwell-native 4-bit floating-point format based on E2M1 representation (NVIDIA, 2024). Its finite magnitudes are

$$\mathcal{F}_{\mathrm{E2M1}} = \{0, \pm 0.5, \pm 1, \pm 1.5, \pm 2, \pm 3, \pm 4, \pm 6\}. \tag{19}$$

To preserve dynamic range at low precision, NVFP4 employs hierarchical scaling. Tensor elements are partitioned into micro-blocks of 16 values along the quantization axis. Each block shares an FP8 E4M3 scale, while a second FP32 scale is shared by the tensor.

Let X_g denote a 16-element quantization block of a high-precision tensor X. We denote its tensor-level FP32 scale by α_X and its block-level E4M3 scale by β_g. Conceptually, NVFP4 quantization and reconstruction can be written as

$$\widehat{X}_g = \mathrm{Round}_{\mathrm{E2M1}}\left(\frac{X_g}{\alpha_X \beta_g}\right), \qquad \widetilde{X}_g = \alpha_X \beta_g \widehat{X}_g, \tag{20}$$

where $\widehat{X}_g$ denotes the packed E2M1 representation and $\widetilde{X}_g$ its reconstructed value.

For the standard max-based NVFP4 recipe, let

$$A_g = \max_{x \in X_g} |x| \tag{21}$$

denote the maximum magnitude within block g. Since the largest E2M1 magnitude is 6, the local scale is chosen so that the normalized block approximately spans the available E2M1 range:

$$\beta_g \approx \mathrm{Round}_{\mathrm{E4M3}}\left(\frac{A_g}{6\alpha_X}\right). \tag{22}$$

The two-level scaling allows each 16-element block to adapt to its local dynamic range while the tensor-level FP32 scale extends the overall representable range.

In the main text, we use $\mathcal{Q}_{\mathrm{NVFP4}}(\cdot)$ to denote this quantization procedure, including the E2M1 values and their associated scales. A superscript specifies the quantization layout when necessary; for example, $\mathcal{Q}^{\mathrm{hd16}}_{\mathrm{NVFP4}}(\cdot)$ denotes independent 16-element NVFP4 blocks along the head dimension, as used by the persistent KV cache.

For Q/K quantization, ARGUS additionally adopts Four-over-Six (4/6) scaling (Cook et al., 2025), which evaluates target magnitudes $\rho \in \{4, 6\}$ instead of always mapping the block maximum to 6. This choice is also used to derive the quantization-aware sparse-centering threshold in Appendix B.1.1.

B Additional Method Derivations

B.1 Softmax Equivalence of Calibrated Centering

Recall

$$\widetilde{Q} = Q - \mathbf{1}_{N_q}\mu_Q^\top, \qquad \widetilde{K} = K - \mathbf{1}_{N_k}\mu_K^\top. \tag{23}$$

ARGUS evaluates the compensated score

$$\widetilde{S} = \frac{\widetilde{Q}\widetilde{K}^\top + \mathbf{1}_{N_q}(\widetilde{K}\mu_Q)^\top}{\sqrt{d}}. \tag{24}$$

Expanding the numerator gives

$$\widetilde{Q}\widetilde{K}^\top + \mathbf{1}_{N_q}(\widetilde{K}\mu_Q)^\top = QK^\top - (Q\mu_K)\mathbf{1}_{N_k}^\top. \tag{25}$$

The remaining term is constant within each query row and therefore vanishes under softmax. Hence,

$$\mathrm{softmax}(\widetilde{S}) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right). \tag{26}$$

Thus, calibrated centering changes the numerical representation presented to quantization without changing the corresponding full-precision attention probabilities.

B.1.1 FP4 Resolution for Sparse Centering

Consider a quantization group with maximum magnitude A_g, whose maximum is scaled to a target magnitude S. The corresponding scale is

$$s_g(S) = \frac{A_g}{S}. \tag{27}$$

E2M1 has a smallest positive representable value of 0.5. Its zero-rounding boundary therefore occurs at half of this value, giving

$$\tau_g(S) = 0.25\, s_g(S) = \frac{A_g}{4S}. \tag{28}$$

For a set of candidate target magnitudes, the finest available resolution is determined by the largest target $S_{\max}$:

$$\tau_g^{\mathrm{FP4}} = \frac{A_g}{4S_{\max}}. \tag{29}$$

ARGUS uses Four-over-Six (4/6) scaling for Q/K (Cook et al., 2025), with $S \in \{4, 6\}$. Hence $S_{\max} = 6$, yielding

$$\frac{\tau_g^{\mathrm{FP4}}}{A_g} = \frac{1}{24}. \tag{30}$$

Table 6: Generation settings used for accuracy evaluation. GSM8K, RULER, and LongBench use greedy decoding with one output per example. For AIME25 and GPQA-Diamond, we sample multiple responses per problem and report accuracy averaged over all sampled responses. All methods use the same decoding configuration and base seed.

Benchmark	Decoding	Samples per problem	Temp.	p	k	$p_{\min}$	Max new tokens	Base seed
GSM8K	Greedy	1	0	1.0	–	–	512	–
RULER	Greedy	1	0	1.0	–	–	Task-specific 30–128	–
LongBench-v1	Greedy	1	0	1.0	–	–	Task-specific 16–512	–
LongBench-v2	Greedy	1	0	1.0	–	–	128	–
AIME25	Sampling	8	0.6	0.95	20	0	38,912	42
GPQA-Diamond	Sampling	4	0.6	0.95	20	0	38,912	42

For the sampling-based evaluations, we use zero-shot prompting, presence penalty $= 0$, and repetition penalty $= 1$. AIME25 contains 30 problems and therefore produces $30 \times 8 = 240$ responses per method; GPQA-Diamond contains 198 problems and produces $198 \times 4 = 792$ responses per method. RULER uses the official task-dependent output budgets; the budget is clipped only when necessary to remain within the model context window. LongBench-v2 uses the official 128-token direct-answer protocol.

Equivalently, the two target magnitudes produce relative zero-rounding boundaries $1/16$ and $1/24$, respectively, and the finer resolution is $1/24$.

ARGUS uses this FP4 resolution to construct the sparse centering mask from the offline-calibrated channel offsets, as described in Section 3.1.

C Baseline implementations

We integrated SageAttention3 and ThriftAttention into the same HF/SGLang evaluation framework without altering their quantization formats, sparsity policies, or attention formulations. For SageAttention3, we added paged-BF16 KV-cache integration, compact GQA layout conversion, dynamic-length handling, and chunked-prefill support while preserving its original numerical behavior. For ThriftAttention, we implemented its mixed BF16–NVFP4 cache in a paged layout with incremental updates and corrected short-sequence mapping, tail masking, cache-slot reuse, CUDA-stream handling, and chunked-prefill alignment. All adaptations were validated through operator-level and end-to-end generation tests; therefore, the reported results retain the semantics of the original methods.

D Evaluation settings

Models and Benchmarks. We evaluate ARGUS on representative GQA language models from the Qwen3, Llama-3, and Ministral-3 families (Yang et al., 2025; Grattafiori et al., 2024; Liu et al., 2026). We consider both short- and long-context workloads. GSM8K (Cobbe et al., 2021) evaluates mathematical reasoning, while AIME25 and GPQA evaluate more challenging reasoning capabilities. For long-context evaluation, we use LongBench (Bai et al., 2024a), LongBench-v2 (Bai et al., 2024b), and RULER (Hsieh et al., 2024). Additional language-modeling evaluations on WikiText-2 and C4, together with downstream evaluations using the LM Evaluation Harness (Gao et al., 2024a), are reported in Table 10 in the appendix. LongBench-v1/v2 and GSM8K are evaluated through the Transformers backend, whereas RULER, AIME25, and GPQA use SGLang. For Qwen3 on AIME25 and GPQA, we enable its native thinking mode. The detailed cross-four-model evaluations are reported in Table 7

Generation Settings. We use greedy decoding for GSM8K, RULER, and LongBench v1/v2 (Cobbe et al., 2021; Bai et al., 2024a;b). For AIME25 and GPQA (Zhang & Math-AI, 2025; Rein et al., 2024), each question is evaluated with multiple generated samples under identical decoding

Table 7: GSM8K accuracy (%) ↑. Bold indicates the best low-precision result.

Method	Qwen3-8B	Qwen3-4B	Llama-3.1-8B	Ministral-3-8B
BF16	89.08	85.37	32.60	92.27
SageAttention3	85.97	83.70	24.19	91.43
ThriftAttention	85.44	**84.00**	25.47	90.67
ARGUS	**88.93**	83.93	**26.31**	**91.51**

configurations, including the same temperature, seed policy, and other generation parameters across methods. This evaluation measures robustness across multiple sampled responses per problem under a shared generation protocol. Detailed generation settings are summarized in Table 6.

Model Settings. We evaluate long-context performance using two 8B-scale models, thinking mode is enabled, we use Qwen3-4B and Qwen3-8B to evaluate the impact of low-precision attention under reasoning-intensive workloads. We use Llama-3.1-8B-Instruct for the long-context benchmarks, while Llama-3.1-8B is used for the remaining evaluations.

Accuracy Metrics. Following SageAttention3 (Zhang et al., 2026a), we evaluate the numerical accuracy of the quantized attention output O' against the BF16 reference output O using relative L1 error, mean squared error (MSE), and cosine similarity. We flatten O and O' into vectors with n elements and compute

$$\mathrm{L1} = \frac{\|O - O'\|_1}{\|O\|_1}, \qquad \mathrm{MSE} = \frac{1}{n}\|O - O'\|_2^2, \qquad \mathrm{CosSim} = \frac{O^\top O'}{\|O\|_2 \|O'\|_2}. \tag{31}$$

Lower L1 and MSE and higher cosine similarity indicate better agreement with the BF16 reference.

E Exploratory Low-Precision Decode Variants

We further explore whether decoding can employ more aggressive low-precision computation beyond the NVFP4 KV-cache design used in the main method. We consider two additional variants, ARGUS-D48 and ARGUS-UP. Their results on LongBench-v2 and AIME25 are summarized in Tables 8 and 9.

ARGUS-D48. ARGUS-D48 explores a more aggressive mixed-precision decode path. Prefill follows the low-precision operator used by ARGUS. During decoding, Q and K employ sparse centering followed by 16-channel NVFP4 4-over-6 quantization, together with a BF16 correction sidecar to compensate for the score bias introduced by low-precision centering. The probability path and PV multiplication use FP8, with the softmax normalization computed from the reconstructed low-precision probability representation. This variant is designed to study whether an NVFP4 QK path combined with FP8 P/V computation can further reduce decode precision without substantial quality degradation.

ARGUS-UP. ARGUS-UP is an accuracy-oriented reference variant. Both prefill and decoding apply a normalized H_{128} Hadamard transform to each Q/K head and execute the QK centering online in the transformed domain. During decoding, newly generated Q/K vectors undergo the same transform, while K is centered using the mean obtained during prefill to maintain a consistent representation across stages. Explicit Q centering is unnecessary because the corresponding term adds the same constant to all attention logits of a query and therefore cancels under softmax. K is stored in a centered NVFP4 cache, while V uses an uncentered NVFP4 cache, and the decode attention computation remains in BF16.

Discussion. These more aggressive decode variants do not provide consistent accuracy improvements across models and tasks. For example, ARGUS-D48 performs strongly on Llama-3.1-8B-Instruct on LongBench-v2, but degrades noticeably on Qwen3-8B, while none of the variants consistently improves over BF16 on AIME25. We therefore retain NVFP4 KV caching with BF16

Table 8: LongBench-v2 results of exploratory decode variants. Bold indicates the best overall score within each model. Higher is better for all task scores (↑).

Model	Method	Overall	Easy	Hard	Short	Medium	Long
Qwen3-8B	BF16	**34.19**	37.50	32.15	40.56	31.16	29.63
	ARGUS	31.21	38.02	27.01	36.67	26.05	32.41
	ARGUS-D48	28.03	30.21	26.69	36.11	21.86	26.85
	ARGUS-UP	30.62	37.50	26.37	36.67	27.91	25.93
Llama-3.1-8B-Inst.	BF16	29.03	29.69	28.62	32.78	27.44	25.93
	ARGUS	30.22	28.65	31.19	34.44	29.77	24.07
	ARGUS-D48	**31.41**	29.69	32.48	35.56	30.70	25.93
	ARGUS-UP	31.21	31.25	31.19	36.67	28.37	27.78

Table 9: Mean accuracy on AIME25 for exploratory decode variants.

Method	Mean Accuracy↑
BF16 / FlashInfer	65.00%
ARGUS	63.33%
ARGUS-D48	63.33%
ARGUS-UP	62.50%

decode computation as the default ARGUS design, and leave more aggressive mixed-precision decoding for future work.

F Additional Evaluations

Additional Language Modeling and Downstream Evaluation. Table 10 evaluates language-modeling performance using perplexity on WikiText-2 and C4, together with downstream task accuracy using the LM Evaluation Harness (Gao et al., 2024a). Across the three models, SageAttention3 generally achieves the lowest perplexity among the low-precision methods, while ThriftAttention often provides stronger downstream accuracy, particularly on Qwen3-8B and Llama-3.1-8B. ARGUS is not uniformly the best on every individual metric, but remains competitive across both evaluation settings and broadly close to the BF16 reference. Notably, on Ministral-3-8B, ARGUS achieves the highest average LM-Eval accuracy among all evaluated methods. Together with the operator-level and end-to-end speedups reported in the main text, these results demonstrate a favorable accuracy–efficiency trade-off across different models and evaluation objectives.

Additional Efficiency Results on Prefill Operator. Table 11 illustrates the benefit of Cache-Ready Prefill. ARGUS substantially reduces the fraction of preprocessing and KV-cache materialization in the complete prefill operator across all evaluated sequence lengths. Although this fraction decreases for all methods as sequence length grows and the main attention computation becomes dominant, ARGUS consistently keeps the auxiliary overhead much smaller than SageAttention3 and ThriftAttention.

KV-cache capacity. NVFP4 increases the single-GPU serving capacity of Qwen3-4B by $3.0\times$–$3.5\times$ over the BF16 KV cache. We evaluate the maximum sustainable concurrency at input lengths from 16K to 128K on a single NVIDIA RTX 5090. Both backends use BF16 model weights, generate 128 tokens per request, and share the same 16K global prefill budget. CUDA Graph, overlap scheduling, and radix caching are disabled to isolate the KV-cache capacity. Every reported boundary completes the requested generation and reaches the target decode batch. As shown in Table 12, NVFP4 supports $3.5\times$ more concurrent requests at 16K, 32K, and 64K, and $3.0\times$ more requests at 128K. The lower gain at 128K is caused by integer concurrency granularity: the automatically profiled token-pool capacity increases from 134,656 tokens with BF16 to 479,104 tokens with NVFP4, corresponding to a $3.558\times$ increase.

Table 10: Perplexity and LM-Eval results across different GQA models. Lower is better for PPL, while higher is better for LM-Eval metrics. Wiki denotes WikiText-2; C4 denotes the Colossal Clean Crawled Corpus; ARC-C and ARC-E denote ARC-Challenge and ARC-Easy; HellaS denotes HellaSwag; LAMB denotes LAMBADA; PIQA denotes Physical Interaction: Question Answering; and Wino denotes WinoGrande.

		PPL ↓		LM-Eval ↑						
Model	**Method**	**Wiki**	**C4**	**ARC-C**	**ARC-E**	**HellaS-**	**LAMB-**	**PIQA**	**Wino-**	**Avg.**
Qwen3-8B	BF16	9.7245	13.9400	56.31	80.85	74.87	64.04	77.69	68.27	70.34
	Sage3	9.6453	13.9464	55.03	78.75	75.26	63.26	76.99	66.77	69.34
	Thrift	9.9535	14.4569	56.40	80.56	75.23	63.26	77.53	67.80	70.13
	ARGUS	9.7327	14.0507	56.23	81.06	74.27	64.16	77.97	68.82	**70.42**
Llama-3.1-8B	BF16	6.2406	9.2617	53.84	81.10	78.93	75.35	81.28	73.64	74.02
	Sage3	6.4088	9.4914	52.22	77.02	78.64	74.40	79.33	69.46	71.84
	Thrift	6.7118	10.0127	53.92	81.23	78.91	74.69	81.07	74.11	73.99
	ARGUS	6.5545	9.7642	50.77	77.57	78.47	75.37	80.79	70.88	72.31
Ministral-3-8B	BF16	6.9113	10.1375	65.36	85.44	78.56	71.57	80.69	73.01	75.77
	Sage3	6.9808	10.2081	63.74	84.72	78.47	71.45	79.54	71.51	74.91
	Thrift	7.3477	10.6980	65.27	85.40	78.43	71.16	80.79	72.45	75.58
	ARGUS	6.9905	10.2255	63.23	85.06	78.05	71.45	80.20	72.53	75.09

Table 11: Fraction of preprocessing and KV-cache materialization in the complete prefill operator latency (%) ↓ across sequence lengths. Lower is better.

Method	**1K**	**2K**	**4K**	**8K**	**16K**	**32K**
SageAttention3	90.24	78.90	57.75	42.02	29.47	17.70
ThriftAttention	92.54	92.69	63.94	36.41	23.48	14.27
ARGUS	43.50	27.95	13.16	10.10	5.57	2.95

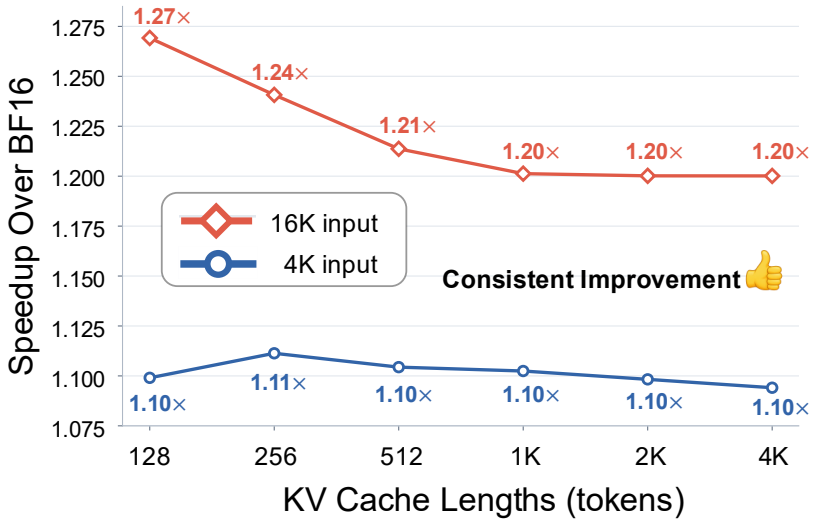


Figure 6: End-to-end SGLang speedup under different decode lengths.

Speedup under different decode lengths. ARGUS maintains a stable end-to-end speedup over the BF16 baseline across a wide range of decode lengths as shown in Figure 6. We evaluate Qwen3-4B under single-request serving ($C = 1$), fixing the input length to either 4K or 16K tokens and

Table 12: Single-GPU KV-cache capacity for Qwen3-4B. Total input tokens are calculated as the input length multiplied by the maximum successfully served concurrency.

Input	Maximum concurrency		Total input tokens		Capacity
length	BF16	NVFP4	BF16	NVFP4	gain
16K	8	28	131,072	458,752	3.50×
32K	4	14	131,072	458,752	3.50×
64K	2	7	131,072	458,752	3.50×
128K	1	3	131,072	393,216	3.00×

Table 13: Average LongBench score and speedup. Higher is better for both metrics (↑).

Method	**Avg.↑**	**Speedup↑**
Llama-3.1-8B-Instruct		
BF16	**17.51**	1.00×
Sage3	15.21	0.52×
Thrift	13.54	0.74×
ARGUS	**15.07**	**1.13**×
Qwen3-8B		
BF16	23.24	1.00×
Sage3	20.64	–
Thrift	19.65	–
ARGUS	**22.70**	**1.14**×

varying the number of generated tokens from 128 to 4,096. For the 4K input, the speedup remains within 1.09×–1.11×, while the 16K input achieves 1.20×–1.27×. The speedup decreases moderately as the decode length increases, particularly for the 16K input, where it declines from 1.27× at 128 generated tokens to approximately 1.20× at 4,096 tokens. This trend is expected because longer generation amortizes the fixed prefill advantage and increases the contribution of non-attention model operations during autoregressive decoding. Nevertheless, the speedup converges to a stable positive margin rather than vanishing, demonstrating that Argus preserves its end-to-end benefit even for long decode workloads.

Absolute end-to-end serving latency. Argus provides increasingly pronounced end-to-end acceleration as the input context grows, reaching a 1.620× E2E speedup at 64K tokens. Table 14 reports the absolute TTFT, TPOT, and E2E latency of Qwen3-8B for single-request serving ($C = 1$) with exactly 128 generated tokens. Both systems use BF16 model weights, while the baseline stores the KV cache in BF16 and Argus uses its NVFP4 KV-cache path. For each configuration, we launch three independent server instances, perform a same-length warm-up, and issue four timed requests per server instance; the reported value is the median of the per-instance request medians. The 4K–32K experiments use the model's native 40,960-token context configuration. The 64K experiment uses identical YaRN and chunked-prefill settings for both backends. Argus improves TTFT from 1.069× at 4K to 1.379× at 32K and 1.708× at 64K. TPOT acceleration similarly increases from 1.058× to 1.184× and 1.304×, respectively. Consequently, the overall E2E speedup grows from 1.059× at 4K to 1.307× at 32K and 1.620× at 64K, demonstrating that the benefits of the compressed attention path become more significant as attention accounts for a larger fraction of serving latency.

End-to-end latency speedup on the Transformers backend. The results are shown in Table 13.

Table 14: End-to-end serving latency of Qwen3-8B under single-request concurrency ($C = 1$) with exactly 128 generated tokens. TTFT and E2E latency are reported in milliseconds, while TPOT is reported in milliseconds per token. Speedup is calculated as BF16 divided by ARGUS.

Input	**TTFT (ms)**			**TPOT (ms/token)**			**E2E (ms)**		
	BF16	**ARGUS**	**Sp.↑**	**BF16**	**ARGUS**	**Sp.↑**	**BF16**	**ARGUS**	**Sp.↑**
4K	324.18	303.25	1.069×	11.544	10.916	1.058×	1790.35	1690.02	1.059×
8K	655.82	583.11	1.125×	11.812	11.115	1.063×	2155.07	1996.31	1.080×
16K	1447.76	1187.46	1.219×	12.563	11.422	1.100×	3043.01	2637.31	1.154×
32K	3526.57	2557.40	1.379×	14.257	12.040	1.184×	5339.23	4086.51	1.307×
64K (YaRN)	10387.57	6082.99	**1.708**×	17.285	13.258	**1.304**×	12582.44	7764.84	**1.620**×

Table 15: Inference-friendly calibration ablations. Each entry reports mean / worst-case accuracy across evaluated layers. LR Scale means learnable scale which can be merged in q/k norm.

Configuration	**L1↓**	**MSE↓**	**CosSim↑**
Q-Smooth	0.1608 / 0.2044	5.12e-3 / 4.46e-2	0.9873 / 0.9813
Q-Smooth + Offline Stat.	0.1625 / 0.2073	5.25e-3 / 4.56e-2	0.9871 / 0.9808
K-Smooth	0.2160 / 0.3497	1.48e-2 / 1.81e-1	0.9752 / 0.9429
K-Smooth + Offline Stat.	–	1.50e-2 / 1.90e-1	0.9750 / 0.9317
K-Smooth + LR Scale (Cos.)	0.2089 / 0.2969	1.20e-2 / 1.21e-1	0.9772 / 0.9594
K-Smooth + LR Scale (Const.)	0.2120 / 0.2908	1.21e-2 / 1.20e-1	0.9764 / 0.9610
QK-Smooth	0.1507 / 0.1955	4.77e-3 / 4.34e-2	0.9890 / 0.9833
QK-Smooth + LR Scale (Cos.)	0.1475 / 0.1897	4.50e-3 / 3.98e-2	0.9895 / 0.9844
QK-Smooth + LR Scale (Const.)	0.1472 / 0.1905	4.53e-3 / 4.11e-2	0.9896 / 0.9841

G Additional Inference-Friendly Ablations

Beyond the configurations reported in the main text, we explored several additional calibration strategies that introduce no or negligible runtime overhead. Table 15 compares QK smoothing, offline statistics, and calibrated scaling variants. Overall, jointly calibrating Q and K is substantially more effective than modifying either operand alone, while additional offline scale calibration further reduces the output error.

Empirically, we observe that the locations of persistent outlier channels strongly align with channels having large Q/K normalization weights, as shown in Figure 7. This suggests that part of the channel-wise magnitude imbalance is already encoded in the Q/K normalization parameters. Motivated by this observation, we absorb the calibrated per-channel scaling factors into the Q/K normalization weights, eliminating additional runtime scaling operations. To preserve the structure induced by RoPE, we further tie the scaling factors within each RoPE channel pair.

NVINT4 vs. NVFP4. We also explored E0M3 (NVINT4) using the `mma.mxfpnvfp` (NVIDIA Corporation, 2026b) instruction, selecting between NVINT4 and NVFP4 for Q, K, P, and V according to per-tensor reconstruction MSE. Only V occasionally favored NVINT4, but its slightly lower reconstruction error did not translate into better attention outputs. Among the 25 layers where NVINT4 was selected for V, output MSE improved in only 9 layers and degraded in 16, with the aggregate output MSE worsening by 0.415%. As expected, the P metrics remain unchanged because V is applied after softmax. We therefore retain NVFP4 for V in the final design.

H Hardware and Workload Sensitivity

We assess where ARGUS should provide the largest benefit by combining published kernel measurements with complete-operator and request-level latency accounting. RTX 5090 is experimentally validated and conclusions for other devices are predictions about an architecture-adapted implementation.

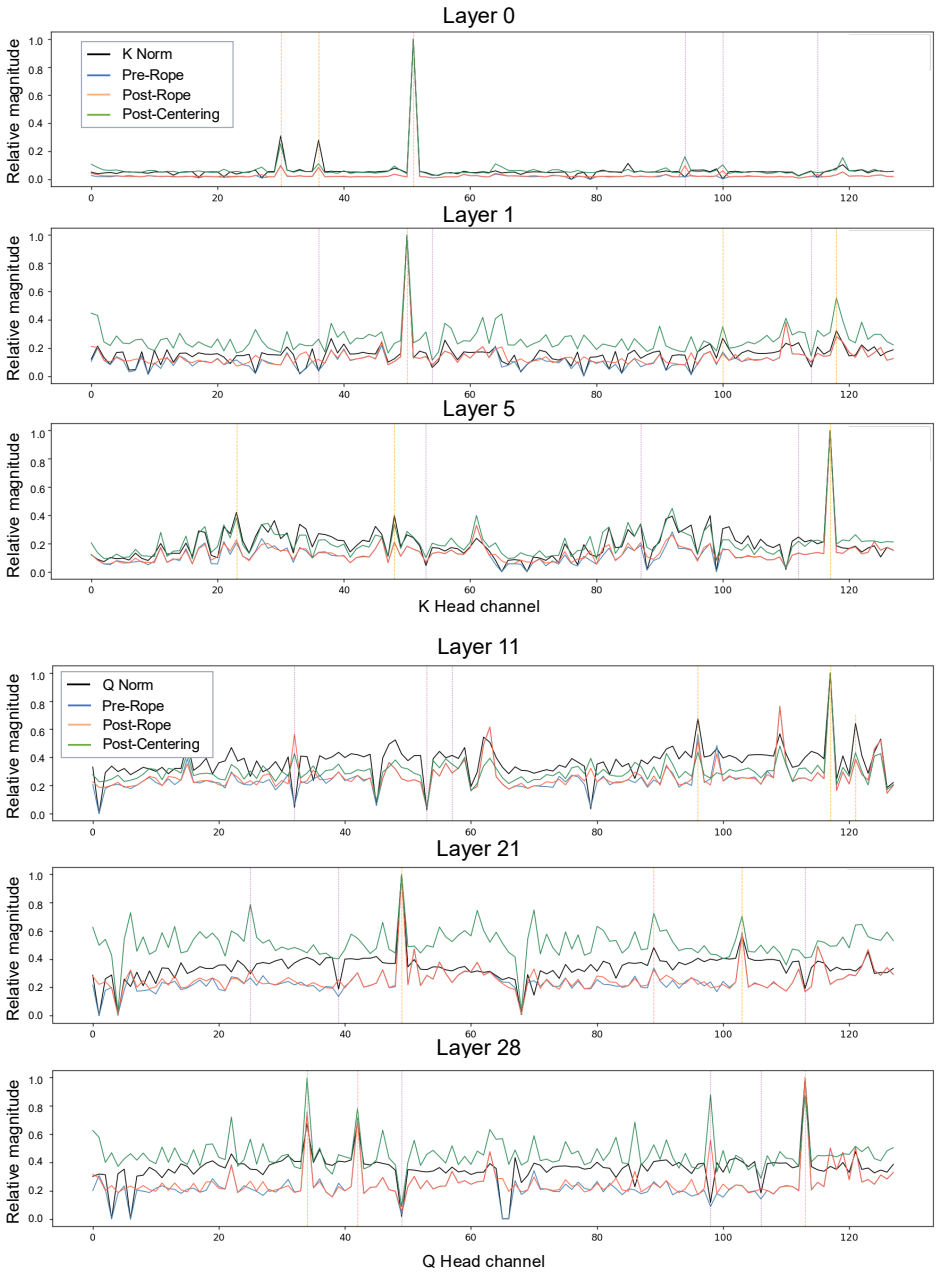


Figure 7: Layer-wise Q/K channel magnitude statistics under different processing stages.

H.1 From operator acceleration to request latency

For an isolated request, decompose baseline latency as $T_0 = T_{p,0} + T_{d,0} + T_{o,0}$, where $T_{p,0}$ is prefill attention, $T_{d,0}$ is cumulative decode attention, and $T_{o,0}$ is all remaining work, including projections, MLPs, normalization, sampling, and framework costs. Attention times sum over layers and include preprocessing and persistent-cache construction or updates. Define the baseline time fractions and ARGUS operator speedups by

$$f_p = \frac{T_{p,0}}{T_0}, \qquad f_d = \frac{T_{d,0}}{T_0}, \qquad s_p = \frac{T_{p,0}}{T_{p,A}}, \qquad s_d = \frac{T_{d,0}}{T_{d,A}}.$$

Here A denotes ARGUS. With $\epsilon = (T_{o,A} - T_{o,0})/T_0$, adding the accelerated times gives

$$T_A = \frac{T_{p,0}}{s_p} + \frac{T_{d,0}}{s_d} + T_{o,0} + \epsilon T_0, \tag{32}$$

$$S_{\text{e2e}} = \left(1 - f_p - f_d + \frac{f_p}{s_p} + \frac{f_d}{s_d} + \epsilon\right)^{-1}. \tag{33}$$

At fixed $f_p + f_d$, s_p, s_d, and ϵ, increasing f_p improves speedup if $s_p > s_d$. Increasing generation length changes both the stage mix and cache lengths, so it need not change speedup monotonically. A small $f_p + f_d$ limits the benefit regardless of operator speed: for $\epsilon = 0$, even eliminating attention entirely gives at most $1/(1 - f_p - f_d)$. The fractions must describe the target device and workload; concurrent execution requires critical-path accounting rather than summing overlapping kernel durations.

H.2 Hardware parameters and their implications

Table 16: Peak hardware parameters. Tensor throughput is dense PFLOP/s; bandwidth is local GPU-memory TB/s. System values are normalized per GPU. Sources: RTX 5090 (NVIDIA Corporation, 2025b), RTX PRO 6000 (NVIDIA Corporation, 2025c), Thor (NVIDIA Corporation, 2026a), B200 (NVIDIA Corporation, 2024a; Zadouri et al., 2026), GB200 (NVIDIA Corporation, 2024b), and GB300 (NVIDIA Corporation, 2025a).

Device	BF16	FP4	Bandwidth	FP4/BF16
RTX 5090	0.2095	1.676	1.792	$8\times$
RTX PRO 6000 (WS)	0.5038	2.0152	1.792	$4\times$
Jetson T5000 (MAXN)	—	1.035	0.273	—
B200	2.2500	9.000	8.000	$4\times$
GB200 GPU	2.5000	10.000	8.000	$4\times$
GB300 GPU	2.5000	15.000	8.000	$6\times$

RTX 5090 and RTX PRO 6000 BF16 rates use FP32 accumulation. WS denotes the Blackwell Workstation Edition. ThriftAttention reports a 96 GB RTX PRO 6000 Blackwell GPU without specifying its exact edition (Sharratt, 2026). The cited T5000 specification does not list a matching BF16 rate. CPU-memory and NVLink bandwidth are not included as local GPU-memory bandwidth. Compared with RTX 5090, RTX PRO 6000 increases BF16 and FP4 throughput by approximately $2.4\times$ and $1.2\times$, respectively, at unchanged bandwidth. Relative to RTX 5090, B200 increases BF16 throughput by approximately $10.7\times$ but memory bandwidth by $4.46\times$; stronger compute therefore does not preserve the same compute–memory balance. The FP4/BF16 ratios describe matrix-throughput headroom, not operator speedups. Data-center prefill also depends on probability construction and on-chip movement (Zadouri et al., 2026); GB300 doubles the exponential throughput of B200/GB200, without removing these other costs (Aubrey & Stam, 2025).

H.3 Prefill: empirical reference and operator estimate

For hardware h and implementation $x \in \{0, A\}$, let $T^x_{\text{attn},h}$ be main-kernel latency and $T^x_{\text{aux},h}$ the exposed work outside that measurement: centering, quantization, layout conversion, and persistent-cache construction. Internal softmax and data movement are already timed in the main kernel. A hat denotes a conditional estimate:

$$\widehat{s}_{p,h} = \frac{T^0_{\text{attn},h} + T^0_{\text{aux},h}}{\widehat{T}^A_{\text{attn},h} + \widehat{T}^A_{\text{aux},h}}. \tag{34}$$

This accounts for the complete operator rather than applying an independent softmax speedup multiplier to a published kernel result.

These measurements support roughly 1.6–$1.9\times$ as an optimistic *kernel reference* at the listed shapes. The same source's higher-fidelity native NVFP4/NVFP4 controls achieve only $0.846\times$ and $0.951\times$ (cosine approximately 0.982). Transferring the faster reference to ARGUS therefore requires recovering acceptable accuracy and adapting noncausal MHA execution to causal GQA; matching

Table 17: GB200 kernel references from Hu (2026, Table 25). $B = 1$, $H_q = H_{kv} = 64$, $d = 128$, noncausal, prequantized inputs. Times are μs. The low-precision rows use the TK NVFP4/NVFP4 fixed schedule.

N	BF16	NVFP4/NVFP4	Kernel Speedup↑	CosSim↑
4096	370.688	227.328	1.631×	0.962449
8192	1611.488	861.792	1.870×	0.962801

operand formats alone is insufficient. VC-Attention independently identifies probability construction as a bottleneck, but its ExpCast-FP8 is not an NVFP4 encoder (Li et al., 2026).

For a target operator speedup $S_\star$, rearranging the estimate gives

$$\widehat{T}^A_{\mathrm{aux},h} - \frac{T^0_{\mathrm{aux},h}}{S_\star} \le \frac{T^0_{\mathrm{attn},h}}{S_\star} - \widehat{T}^A_{\mathrm{attn},h}. \tag{35}$$

At $S_\star = 1.5$, the two reference pairs leave respective budgets of $19.80\,\mu$s and $212.53\,\mu$s on the right-hand side. These are allowable costs derived from measurements, not assumed ARGUS costs. They motivate a testable $1.5\times$ operator target for a successful backend transfer. As auxiliary work becomes small, the estimate approaches the paired kernel ratio. Thus, *data-center prefill gains require both an efficient probability path and low external preparation costs*; ARGUS addresses the latter and must be combined with a suitable target-device kernel for the former.

H.4 Decode: cache savings and cumulative latency

ARGUS retains BF16 decode arithmetic. Its NVFP4 cache uses $b_4 = 4/8 + 1/16 = 9/16$ byte per element, including block scales, compared with $b_{16} = 2$ for BF16. Thus $r_{\mathrm{KV}} = b_{16}/b_4 = 32/9$ before tensor scales, padding, and page metadata. In a streaming regime with comparable effective bandwidth and unchanged remaining work, net acceleration requires

$$T_{\mathrm{convert}} < \left(1 - \frac{1}{r_{\mathrm{KV}}}\right) \frac{M_{d,0}}{W^{\mathrm{eff}}}, \tag{36}$$

where $M_{d,0}$ is baseline KV-read traffic, W^{eff} is achieved external-memory bandwidth, and T_{convert} includes the additional, unhidden conversion and cache-update cost over the same interval. Lower bandwidth increases the time saved by removing a fixed number of bytes, but not the ideal traffic-only speedup, which is r_{KV} on every device. This identifies KV-heavy generation with efficiently hidden conversion as Thor's most promising ARGUS regime.

To obtain a request-level decode estimate, sum matched complete-operator latencies rather than average individual speedups:

$$\widehat{s}_d = \frac{\sum_{t=0}^{G-1} \sum_\ell T^{(\ell,t)}_{d,0}}{\sum_{t=0}^{G-1} \sum_\ell \widehat{T}^{(\ell,t)}_{d,A}}. \tag{37}$$

Here ℓ indexes layers and t incremental forward steps; the old cache contains $N + t$ tokens. If prefill produces the first output token, $G = D - 1$ for D outputs. Timings include current-token work and cache updates. Our RTX 5090 measurements supply matched inputs; target-device estimates require equivalent compute precision, cache layout, and shapes. A longer generation can increase the importance of cache savings, but whole-request benefit still depends on f_d in Eq. 33.

H.5 Expected benefits and optimization priorities

RTX 5090. Long-input prefill and long-cache decoding are both validated opportunities. When attention occupies a comparable request share and $s_p > s_d$, the prefill-weighted workload benefits more. Reducing preprocessing and cache construction remains valuable even after the main kernel is accelerated.

RTX PRO 6000 Blackwell. Compared with RTX 5090, the Workstation Edition combines the same memory bandwidth with a smaller FP4/BF16 throughput ratio and 96 GB of memory (NVIDIA

Corporation, 2025c). We therefore expect useful gains on long-cache workloads, but not automatically larger prefill speedups. ThriftAttention provides supporting evidence: it reports up to $1.7\times$ prefill-kernel speedup over FlashAttention-2 and roughly $1.2\times$ end-to-end prefill speedup at 131k context against FP16 (Sharratt, 2026). These are its mixed FP4/FP16 results, not ARGUS estimates. ARGUS should prioritize low preparation overhead and conversion-efficient cache streaming; larger memory capacity expands feasible workloads without increasing streaming bandwidth.

B200/GB200 and GB300. We expect the most useful prefill gains on long, attention-heavy inputs with an architecture-adapted kernel. B200/GB200 should prioritize probability construction and operand publication; GB300's stronger exponential units make continued optimization of scale delivery, on-chip movement, and pipeline dependencies important. The empirical references support meaningful acceleration, not portable 4–$8\times$ operator gains.

Jetson Thor. The clearest opportunity is long-cache generation in which KV reads are exposed on the critical path. The priority is coalesced cache access and load-path dequantization, not lower decode arithmetic precision.

Overall conclusion. ARGUS is best targeted at attention-dominated long-context inference: optimize the probability/dataflow path for data-center prefill and the compressed-cache load path for bandwidth-constrained decoding. Short-context or weight/MLP-dominated requests should see smaller end-to-end gains. Cache-capacity improvements remain useful but are separate from latency speedup.