RaZeR: Pushing the Limits of NVFP4 Quantization with Redundant Zero Remapping

Yuzong Chen[1]* **Xilai Dai**[*,1] **Jake Hyun**[*,1]
Chi-Chih Chang[1] **Wonsuk Jang**[2] **Yuheng Wu**[2]
Thierry Tambe[2] **Jae-sun Seo**[1] **Mohamed S. Abdelfattah**[1†]

[1]Cornell University [2]Stanford University

Abstract

The recently introduced 4-bit format, NVFP4, demonstrates remarkable performance and memory benefits for quantized large language model (LLM) inference. However, we observe two types of redundancy in NVFP4 encoding: (1) The FP4 element format naturally exposes an unused quantization value due to its sign-magnitude representation that contains both positive and negative zeros. (2) The FP8 block scaling factor has an unused sign bit because it is always positive. Additionally, we find that LLM weights are more tolerant to a lower-precision block scaling factor. Based on these observations, we propose Redundant Zero Remapping (RaZeR), an enhanced numerical format that pushes the limits of NVFP4 for more accurate LLM quantization under the same memory footprint. RaZeR leverages the redundant bits of the block scaling factor to adaptively remap the redundant FP4 zero to additional quantization values with improved accuracy. To demonstrate the practicality of RaZeR, we design efficient GPU kernels for RaZeR-quantized LLM inference and propose novel hardware to natively support this. Extensive experiments validate RaZeR's superior performance for 4-bit LLM quantization. For example, relative to native NVFP4, RaZeR reduces the average perplexity loss by 34.6% and 31.2% under weight-only and weight-activation quantization, respectively. Our work presents a promising direction for designing accurate and efficient low-precision numerics on future hardware platforms. Our code is publicly available at `https://github.com/abdelfattah-lab/NVFP4-RaZeR`.

1 Introduction

The substantial memory bandwidth and computational demands of large language models (LLMs) present critical challenges for their deployment. To tackle this, quantization has emerged as a promising solution for efficient LLM inference. By reducing the operand bit-width, quantization not only decreases the memory footprint, but also improves the computational efficiency on low-precision hardware [18, 2]. While many prior approaches use the standard integer format to represent quantized values [46, 25, 14, 40, 24, 43], recent literature has started to focus on custom low-precision formats that better adapt to LLM tensor distributions, thus reducing the quantization error [38, 33, 9, 11, 21]. Among these, the custom 4-bit NVFP4 format [33], supported by NVIDIA Blackwell GPUs, delivers superior performance for LLM quantization, outperforming earlier 4-bit formats such as INT4 and MXFP4 [13, 2]. On top of FP4, NVFP4 employs block-wise quantization, in which each group of 16 values shares an FP8 scaling factor, with an additional FP32 scaling factor applied per tensor. This fine-grained scaling enables a larger range of tensor values to be represented in FP4 precision while lowering error over prior data formats.

*Equal contribution. The order of the three authors is alphabetical by last name.

†Correspondence to: Mohamed S. Abdelfattah (`mohamed@cornell.edu`).

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

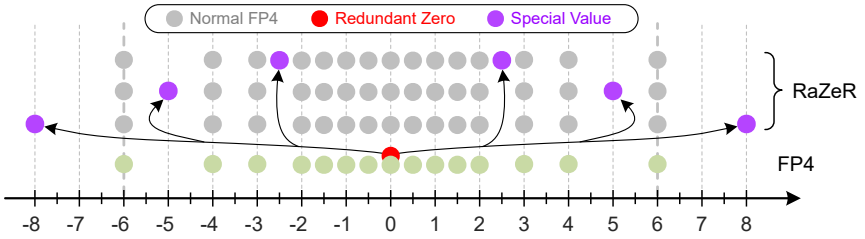


Figure 1: Overview of RaZeR, which remaps the redundant FP4 zero encoding to a special value.

Despite the performance benefits of NVFP4, adopting it for accurate 4-bit weight-only and weight-activation quantization remains challenging due to the limited set of quantization values offered by FP4. In this paper, we propose Redundant Zero Remapping (RaZeR), an enhanced NVFP4 format, illustrated in Fig. 1. FP4 can represent both positive and negative zeros, of which one is redundant. RaZeR adaptively remaps the redundant zero to an additional quantization level, referred to as the *special value*. By carefully selecting this special value, each NVFP4 block can be quantized with the basic FP4 values and a useful special value, thereby reducing per-block quantization error. Additionally, RaZeR exploits redundancy in the block scaling factor to encode special value metadata, thus maintaining the same memory footprint as the baseline NVFP4 format while achieving much higher accuracy.

Beyond algorithmic innovation, we thoroughly study the hardware implications of RaZeR. First, we demonstrate the practicality of deploying RaZeR on current GPU platforms by implementing high-performance kernels to support weight-only quantized LLMs using RaZeR on Nvidia Blackwell GPUs. Second, we design a modified NVFP4 tensor core that efficiently performs multiply-accumulate (MAC) operations when weights and activations are quantized in RaZeR. Our main contributions are summarized as follows:

- We propose RaZeR, an enhanced NVFP4 format that exploits redundancy in NVFP4 block scaling to repurpose the redundant FP4 zero as additional quantization values, enabling more accurate LLM quantization at the same memory footprint.
- We implement efficient GPU kernels that enable RaZeR to be used for 4-bit weight-only quantized LLM inference on Blackwell GPUs, achieving performance comparable to or better than prior kernels.
- We propose a modified NVFP4 tensor core that adds native support for RaZeR with tiny silicon overhead.
- Through comprehensive evaluation, we show that RaZeR outperforms existing 4-bit methods across both weight-only and weight-activation quantization.

2 Related Work

2.1 Quantization Formats

Traditionally, a majority of LLM quantization algorithms rely on the standard integer format [25, 14, 43, 46, 40, 24]. However, LLMs often exhibit outliers that substantially expand the dynamic range, leading to large error under integer quantization. To address this issue, recent work has explored custom quantization formats that better match the distribution of LLM tensors. At 8 bits, ZeroQuant-FP [42] demonstrates that FP8 [32] significantly outperforms INT8 for activation quantization, achieving near-lossless accuracy when combined with INT8 weights. At 4 bits, lookup-based formats such as Normal-Float [9] and Student-Float [11] leverage statistical quantile functions to define the 16 representable values. However, these methods typically store the lookup table in FP16, necessitating high-precision MAC operations with significant energy overhead. In contrast, the FP4-E2M1 format offers a favorable trade-off between model accuracy and hardware efficiency [27, 2], resulting in growing industry adoption and practical impact [35, 33]. The super-range and super-precision formats of Dotzel et al. [11] reassign negative zero to an additional value, while NxFP [29] combines fixed negative-zero remapping with per-block selection between block floating-point and microscaling floating-point formats. Our work further improves FP4 quantization through per-block selection among configurable special values, encoded in redundant NVFP4 block-scale bits, achieving better

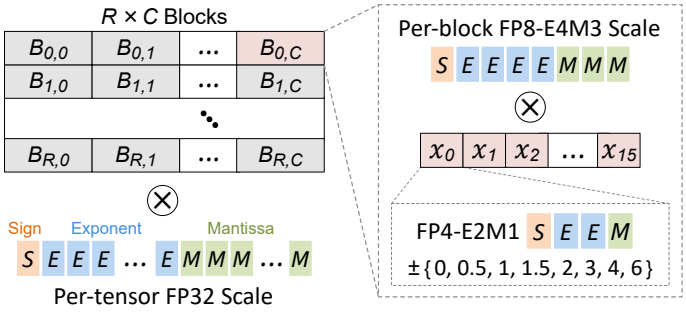


Figure 2: **NVFP4 overview**. A 16-value block of FP4 elements shares an FP8-E4M3 scale, followed by a tensor-wise FP32 scale.

model accuracy without additional storage while remaining compatible with FP4 low-precision hardware.

2.2 Block-wise Quantization

Block-wise quantization is a widely adopted technique to reduce the accuracy loss of quantized models [8, 6]. Prior software-based solutions [46, 25, 26] typically rely on high-precision block scaling factors and slow CUDA cores to perform runtime dequantization. On the other hand, the recent Blackwell GPU natively supports various block-wise quantization formats such as NVFP4 [33], enabling hardware-accelerated scaling under small block sizes (e.g., 16) with improved accuracy.

Motivated by this industry trend, several works have introduced modifications to the NVFP4 quantization algorithm. MR-GPTQ [13] applies GPTQ [14] and Hadamard rotation on top of NVFP4, but finds that adding Hadamard rotation generally hurts model accuracy. FourOverSix [5] improves NVFP4 with adaptive block scaling, where instead of scaling all blocks to the full range of FP4, some blocks can be scaled to a narrower range with less error. Our work follows this line of research and investigates strategies to push the limits of NVFP4 to a new state-of-the-art. Unlike MR-GPTQ and FourOverSix, which refine the NVFP4 *quantization algorithm*, RaZeR modifies the NVFP4 *format* by using redundant block-scale bits to encode per-block special-value selection. Furthermore, we design high-performance GPU kernels and efficient tensor core hardware to demonstrate the practicality of adopting RaZeR on both current and future GPU platforms.

3 Preliminaries on NVFP4

Fig. 2 shows the high-level overview of the NVFP4 format. A tensor is divided into $R \times C$ blocks that share a tensor-wise FP32 scale, and each block contains 16 quantized FP4 values $[x_0, x_1, ..., x_{15}]$ that share an FP8-E4M3[3] scale. Mathematically, the NVFP4 quantization process can be expressed as follows:

$$\Delta^{\text{FP32}} = \frac{|X^{\text{FP32}}|_{\max}}{Q^{\text{FP8}}_{\max} \cdot Q^{\text{FP4}}_{\max}} \tag{1}$$

$$\Delta^{\text{FP8}}_i = \left\lfloor \frac{|X^{\text{FP32}}_i|_{\max}}{\Delta^{\text{FP32}} \cdot Q^{\text{FP4}}_{\max}}, Q^{\text{FP8}} \right\rceil \tag{2}$$

$$\bar{X}^{\text{FP4}}_i = \left\lfloor \frac{X^{\text{FP32}}_i}{\Delta^{\text{FP32}} \cdot \Delta^{\text{FP8}}_i}, Q^{\text{FP4}} \right\rceil \tag{3}$$

where X^{FP32} is the high-precision tensor, $\bar{X}^{\text{FP4}}$ is the quantized FP4 tensor, Q^{FP8} and Q^{FP4} are representable quantization values of FP8 and FP4, Δ^{FP32} is the tensor-wise FP32 scaling factor, Δ^{FP8}_i is the FP8 scaling factor of block i, and $\lfloor \cdot \rceil$ is the rounding function that casts a high-precision FP32 value to a low-precision FP8 or FP4 value.

The rounding function inevitably introduces discrepancy between the original and quantized operands. Consequently, there are two types of quantization error in NVFP4: (1) Rounding of block scaling

[3] In this paper, we use ExMy to represent a format with x-bit exponent and y-bit mantissa.

Table 1: Wikitext-2 and C4 perplexity (↓) under weight-only and activation-only NVFP4 quantization, using different block scale formats. For weight-only quantization, the scale format E3M3 has no accuracy loss compared to the baseline E4M3 format. Results for other models are available in Appendix A.

Scale	**Weight-only NVFP4 Quantization**								**Activation-only NVFP4 Quantization**							
	Llama				Qwen				Llama				Qwen			
	3.1-8B		3.2-3B		3-4B		3-8B		3.1-8B		3.2-3B		3-4B		3-8B	
	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
E4M3	6.63	9.48	8.24	11.10	13.63	16.85	9.92	13.55	6.53	9.34	8.12	10.89	13.81	16.93	9.84	13.49
E4M2	6.67	9.53	8.29	11.17	13.86	16.89	9.99	13.59	**6.59**	**9.43**	**8.18**	**10.98**	**13.94**	**17.02**	**9.92**	**13.58**
E3M3	**6.63**	**9.48**	**8.24**	**11.10**	**13.63**	**16.85**	**9.92**	**13.55**	6.89	9.78	8.43	11.27	14.01	17.26	10.02	13.71
E2M4	6.64	9.50	8.25	11.11	14.13	16.97	10.11	13.65	12.66	20.41	10.83	15.28	14.63	18.16	10.43	14.26
E3M2	6.67	9.53	8.29	11.17	13.96	16.90	9.99	13.59	8.18	11.74	9.50	12.54	14.47	17.88	10.29	14.09
E2M3	6.72	9.63	8.32	11.21	14.22	17.22	10.22	13.79	15.76	26.49	13.13	17.51	15.01	18.47	10.48	14.33

factors to FP8; (2) Rounding of individual element values to FP4. Because FP8 offers much higher precision than FP4, reducing the error caused by FP4 rounding is more effective for improving the overall accuracy of NVFP4, which is the focus of RaZeR.

4 Methodology

In this section, we start by describing two sources of redundancy in the current NVFP4 encoding. We then leverage these redundancies to derive RaZeR, which increases NVFP4 accuracy with no memory overhead.

4.1 Redundancy in NVFP4 Block Scale Encoding

As described in Section 3, NVFP4 quantizes the block scale and the block element to FP8 and FP4, respectively. We observe that both FP8 and FP4 formats have redundancy under the context of NVFP4. According to the standard of [35], the FP8-E4M3 and FP4-E2M1 formats can be expressed as follows:

$$Q^{\text{FP8}} = \begin{cases} (-1)^S \cdot 2^{E-7} \cdot \left(1 + \frac{M}{8}\right) & \text{if } E \neq 0 \\ (-1)^S \cdot 2^{-6} \cdot \frac{M}{8} & \text{if } E = 0 \end{cases} \tag{4}$$

$$Q^{\text{FP4}} = \begin{cases} (-1)^S \cdot 2^{E-1} \cdot \left(1 + \frac{M}{2}\right) & \text{if } E \neq 0 \\ (-1)^S \cdot \frac{M}{2} & \text{if } E = 0 \end{cases} \tag{5}$$

where S, E, and M are sign, exponent, and mantissa of the quantized value, respectively. Based on Eq. 4, the FP8 format contains a sign bit to encode both positive and negative values. However, in Eq. 1 and Eq. 2, both numerator and denominator values are positive, resulting in Δ^{FP32} and Δ_i^{FP8} values that are always positive. Thus, the FP8 block scaling factor used in NVFP4 natively contains a redundant sign bit, resulting in an effective width of 7 bits.

Motivated by this observation, we explore the possibility of further reducing the bit width of block scaling factors. Table 1 left shows the perplexity of Wikitext-2 [30] and C4 [10] datasets across four models from Llama-3 [12] and Qwen-3 [44] families, when applying different block scale formats to NVFP4 weight-only quantization. We find that the E3M3 format has no accuracy drop compared to the baseline E4M3 block scale, while reducing the encoding cost to 6 bits. This can be attributed to the small dynamic range of LLM weights, as observed in prior study [43]. Since the number of exponent bits controls the numerical range, weights can tolerate a scale format with fewer exponent bits. Accordingly, we adopt the E3M3 format for weight block scaling.

We also quantify the impact of different block scale formats on NVFP4 activation quantization, as shown in Table 1 right. We observe that activations are substantially less tolerant to reduced-precision block scales, which can be attributed to their large dynamic range with extreme outliers [7]. This explains why the E4M2 format performs closest to the baseline E4M3 scale. Nevertheless, E4M2 still exhibits noticeable accuracy degradation. Therefore, we choose to retain the 7-bit E4M3 scale format for NVFP4 activation quantization, which offers 1 free bit per block.

4.2 Redundancy in NVFP4 Block Element Encoding

Based on Eq. 5, the FP4 format can represent the set of quantization values $\pm\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$, including both positive zero and negative zero. This redundancy leads to underutilization of the FP4 representational space. Our proposed RaZeR format aims to customize the FP4 format for each block by repurposing the redundant zero as a special value. To determine the per-block special value, RaZeR first defines a set of allowed special values, and for each block, it replaces the redundant zero encoding of FP4 with one special value, such that the block error is minimized. This optimization process can be formulated as:

$$v_i = \underset{v \in V}{\operatorname{argmin}} \left\| \left\lfloor X_i^{\text{Scaled}}, Q^{\text{FP4}} \cup \{v\} \right\rceil - X_i^{\text{Scaled}} \right\|_2^2 \tag{6}$$

$$\bar{X}_i^{\text{RaZeR}} = \left\lfloor X_i^{\text{Scaled}}, Q^{\text{FP4}} \cup \{v_i\} \right\rceil \tag{7}$$

where V is the set of allowed special values, and v_i is the optimal special value for block i, X_i^{Scaled} is the 16-element block after tensor-wise and block-wise scaling, and $\bar{X}_i^{\text{RaZeR}}$ is the quantized RaZeR block. In order to apply Eq. 6, we need to address two key questions: (1) How many special values should be allowed for selection? (2) What values should be included in the set of allowed special values?

To answer the first question, it is important to understand the trade-off between the number of special values and the encoding cost required to select among them. With N different possible special values, selecting one special value per block incurs an encoding overhead of $\lceil \log N \rceil$ bits. As described in Sec. 4.1, the block scales for weight and activation have a redundancy of 2 bits and 1 bit, respectively. Utilizing these redundant bits in the block scales as encoding for special value selection, RaZeR can support 4 special values for weights and 2 for activations, while preserving the same memory footprint as NVFP4. Before deployment, the per-block special value for LLM weights can be precomputed offline without runtime overhead. For activations, however, applying Eq. 6 requires dynamically quantizing each block twice—once for each of the two special values—and comparing their mean squared quantization errors. Fortunately, the overhead of online double quantization is minimal and amounts to less than 2% of the NVFP4 quantizer's runtime, as demonstrated by Cook *et al.* [5].

To answer the second question, it is essential to consider the hardware implication of special values during runtime computation. If a special value can represent arbitrary numbers in FP16 or BF16, the corresponding MAC would need to support high-precision arithmetic. However, this undermines a key benefit of NVFP4—the area and energy efficiency brought by the low-precision arithmetic. Aiming to preserve this benefit, we restrict the special value to be a multiple of 0.5, maintaining the same numerical granularity as FP4 and remaining compatible with low-precision hardware. Additionally, we group the number of allowed special values into pairs, where every pair has the same absolute value but opposite signs. This can further reduce the hardware cost (Sec. 4.4) to support RaZeR. Given these design considerations, the search space of allowed special values is significantly reduced, allowing for a sweep on the possible sets of special values to find the configuration with the lowest quantization error. For weights, this can be done in an offline manner. For activations, we follow prior works [43, 25] and use a calibration set from the Pile dataset [16] to determine the two allowed special values.

Fig. 3 left shows the normalized weight quantization error of four LLMs, when adding one pair of allowed special values to NVFP4. We observe that the quantization error generally follows a parabolic trend as the absolute value increases, with the lowest error occurring at ± 5 across all models. Interestingly, we find that the same trend holds for activation quantization. This is because ± 5 lies at the midpoint between FP4's two largest representable values (i.e., ± 4 and ± 6), and adding it effectively bridges the gap to enable more accurate quantization of near-maximal values. For the weights, we can choose from 4 special values, and hence we measure the quantization error of adding another special value pair on top of ± 5. The optimal set of weight special values for each model is presented in Appendix B.2.

4.3 GPU Kernel Implementation for RaZeR

We demonstrate the feasibility of deploying RaZeR on existing hardware through a weight-only quantized GEMM kernel optimized for Blackwell GPUs [34]. Inspired by the Marlin kernel [15], we adopt a similar execution mechanism to develop an FP4×FP16 kernel with support for redundant zero remapping, targeting high performance across both low-batch and high-batch regimes. We

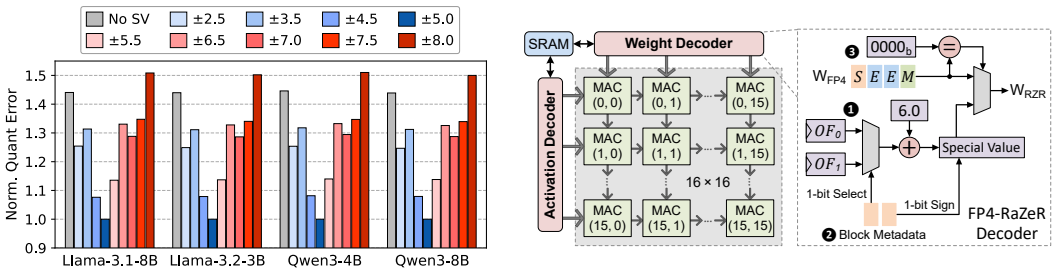


Figure 3: **Left:** Weight quantization error when adding a different pair of special values to FP4. **Right:** Overview of the RaZeR hardware design.

additionally implement a CUDA-core variant to further reduce latency for GEMV workloads beyond the tensor-core implementation. We use a block size of 128 and shuffle the quantized FP4 weights offline so that the load process can load contiguous memory data and exactly match each thread's tensor data-fragment requirements. Every weight block is paired with a single FP16 scale factor for dequantization. Similar to RaZeR for NVFP4, we exploit redundancy in the FP16 scale by using its sign bit and most significant exponent bit to store the 2-bit metadata for selecting special values.

For the tensor-core-based kernel, we employ a work-partitioning strategy similar to Marlin [15]: each SM executes one thread block, and each thread block processes a stripe of weights with approximately equal length, potentially spanning multiple N-dimensions. A final reduction stage aggregates partial results across thread blocks. This ensures that each block has a similar execution time and mitigates tail effects. However, we observed that modern GPUs [34] contain a very large number of streaming multiprocessors (SMs) (e.g., 188 SMs on an RTX Pro 6000). For small weight matrices, this leads to excessive partitioning of the workload, making the sequential reduction stage a non-negligible overhead. To address this issue, we introduce an additional auto-tuning strategy that limits the number of participating SMs and thread blocks, thereby partitioning the weight matrix into larger chunks. The strategy performs offline profiling for each matrix size and, at inference time, selects the optimal number of SMs to launch for each batch size. Although this approach reduces theoretical peak computational throughput, it reduces reduction overhead and lowers GEMM latency for small batch sizes and small weight matrices, achieving up to 9.87% throughput improvement on multiple model inference benchmarks. Appendix E gives a more detailed explanation of the mechanism and profiling data on this optimization.

4.4 Tensor Core Architecture for RaZeR

Since the current NVFP4 tensor cores on Blackwell GPUs do not natively support RaZeR for 4-bit GEMM, we design a custom RaZeR tensor core (Fig. 3 right) to demonstrate its practicality on future GPU platforms. We assume 16×16 MAC units organized in a single-instruction-multiple-data (SIMD) way to align with the block size of NVFP4. The on-chip SRAM sends data to the weight and activation decoders, which convert an FP4 binary input to the corresponding RaZeR value before performing low-precision MAC operations.

Taking the weight decoder as an example, recall from Sec. 4.2 that RaZeR supports four special values, organized into two pairs of additive inverses. To minimize decoder area overhead, we employ an offset-based encoding to store and select these values, as shown in Fig. 3 right. ❶ The weight decoder stores the four special values using two 4-bit offset registers (OF_0 and OF_1). Each register encodes a signed fixed-point value with 1 sign bit, 2 integer bits, and 1 fraction bit, representing values in $[-3.5, 3.5]$ with a step size of 0.5. ❷ At decode time, a 1-bit selector from the metadata chooses the target offset, which is added to 6.0 (the maximum FP4 value) to reconstruct the magnitude. The final special value is obtained by concatenating a 1-bit sign from the metadata. For example, to produce the special value -5.0, an offset register stores 1010_b (i.e., -1.0); adding this offset to 6.0 yields 5.0, which is then combined with a negative sign bit. ❸ During execution, the FP4 weight W_{FP4} is compared against binary zero; upon a match, the decoder outputs the reconstructed RaZeR weight W_{RZR}.

Table 2: Perplexity (↓) under different 4-bit weight-only and weight-activation quantization methods. KV-cache remains in FP16.

#Bits	Method	Llama								Qwen								Avg.
		2-7B		2-13B		3.1-8B		3.2-3B		3-4B		3-8B		3-14B		3-32B		
W-A		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	
16-16	FP16	5.47	6.97	4.88	6.47	6.24	8.96	7.82	10.44	13.66	16.65	9.73	13.30	8.65	12.02	7.61	10.78	9.35
4-16	MXFP4	5.91	7.40	5.17	6.71	7.04	10.05	8.65	11.59	15.01	18.10	10.53	14.22	9.39	12.70	7.93	11.23	10.10
	NVFP4	5.63	7.15	4.98	6.56	6.63	9.48	8.24	11.10	13.83	16.85	9.92	13.55	8.80	12.23	7.87	10.98	9.61
	GPTQ	5.64	7.13	4.97	6.56	6.59	9.48	8.29	11.27	13.96	16.94	10.20	13.68	8.76	12.21	7.75	10.95	9.65
	AWQ	5.60	7.11	4.97	6.55	6.62	9.45	8.23	11.04	14.57	17.59	9.96	13.60	8.87	12.30	7.78	10.97	9.70
	SqueezeLLM	5.61	7.11	4.99	6.56	6.64	9.49	8.24	11.05	14.16	17.27	10.07	13.71	8.98	12.34	7.83	11.02	9.69
	4over6	5.61	7.14	4.97	6.55	6.60	9.42	8.21	11.05	13.83	16.85	9.89	13.54	**8.72**	12.19	7.80	11.04	9.59
	RaZeR	**5.57**	**7.09**	**4.95**	**6.53**	**6.50**	**9.29**	**8.10**	**10.86**	**13.83**	**16.85**	**9.75**	**13.46**	8.77	**12.14**	**7.74**	**10.89**	**9.52**
4-4	MXFP4	7.01	8.67	5.83	7.49	8.29	11.65	10.18	13.49	17.46	20.24	11.84	15.77	10.20	13.62	8.49	11.87	11.38
	NVFP4	5.77	7.29	5.07	6.66	6.95	9.94	8.60	11.65	13.88	17.21	10.05	13.75	8.92	12.38	7.99	11.14	9.83
	NF4	5.83	7.37	5.14	6.70	7.14	10.21	8.87	11.93	14.99	18.07	10.20	14.06	9.16	12.63	8.23	11.62	10.13
	BlockDialect	5.76	7.27	5.08	6.65	6.89	9.87	8.60	11.57	14.48	17.52	10.34	13.99	9.13	12.46	7.92	11.12	9.92
	MR-GPTQ	5.74	7.25	5.07	6.65	6.89	9.92	8.64	12.18	14.11	17.27	10.26	13.85	8.95	12.38	7.87	11.09	9.88
	4over6	5.72	7.25	5.06	6.63	6.88	9.83	8.53	11.53	13.88	17.21	10.00	13.73	**8.83**	12.33	7.91	11.14	9.78
	RaZeR	**5.66**	**7.19**	**5.01**	**6.59**	**6.74**	**9.63**	**8.37**	**11.27**	**13.82**	**17.11**	**9.83**	**13.60**	8.86	**12.26**	**7.84**	**11.02**	**9.68**

5 Evaluations

5.1 Settings

Models. We conduct comprehensive evaluations of RaZeR using models from Llama and Qwen3 families with varying sizes. Specifically, we test Llama-2-{7B, 13B} [41], Llama-3-{3B, 8B} [12], and Qwen3-{4B, 8B, 14B, 32B} [44]. We start from the original HuggingFace Transformers implementations of these models and quantize the model weights using the RaZeR procedure described in Eq. 6–7.

Datasets. We evaluate the perplexity on Wikitext-2 [30] and C4 [10] datasets using a default context length of 2K. We evaluate the accuracy on LM-Eval [17] zero-shot tasks, including PIQA [1], HellaSwag [45], Winogrande [39], ARC [3], and LAMBADA [36]. Additionally, we evaluate two difficult mathematical and logical reasoning tasks, GSM8K [4] and MMLU [19] under chain-of-thought, using the instruction-tuned variant of Llama-3.1-8B and Llama-3.2-3B.

Baselines. We compare RaZeR against previous methods on weight-only and weight-activation quantization. For weight-only quantization, we compare with vanilla MXFP4 [35] and NVFP4 formats, GPTQ [14], AWQ [25], and SqueezeLLM [23]. For weight-activation quantization, we compare with MXFP4, NVFP4, NF4 [9], BlockDialect [21], MR-GPTQ [13], and FourOverSix [5].

Quantization Details. All baseline methods adopt block-wise quantization, except for SqueezeLLM, which only supports per-channel quantization. To promote a fair comparison, we use a block size of 32 with FP16 block scale for GPTQ, AWQ, NF4, and Atom, and a block size of 16 with 8-bit block scale for MXFP4, NVFP4, BlockDialect, MR-GPTQ, FourOverSix, and RaZeR. These configurations ensure that all compared block-wise quantization methods have the same effective precision of 4.5 bits. For RaZeR, we use ± 5 as two allowed special values for both weights and activations, as discussed in Sec. 4.2. The additional two special values for weights vary across different models and are summarized in Appendix B.2.

5.2 Accuracy Results

Perplexity. Table 2 summarizes the perplexity results for weight-only and weight-activation quantization across different methods. RaZeR consistently outperforms prior approaches in both quantization settings, incurring only 0.17 and 0.33 perplexity loss on average, respectively, relative to the FP16 baseline. Compared to two state-of-the-art methods, NVFP4 and FourOverSix, RaZeR reduces the average perplexity loss by 34.6% / 31.2% and 29.2% / 23.3%, respectively, under weight-only / weight-activation quantization. These findings demonstrate that RaZeR can effectively adapt to a multitude of quantization configurations. Additional comparisons on joint quantization of weights, activations, and KV-cache are provided in Appendix C.1.

Table 3: Average zero-shot accuracy (↑) on multiple tasks under different 4-bit weight-activation quantization methods. KV-cache remains in FP16. The detailed results for each task are presented in Appendix C.2.

#Bits W-A	Method	Llama				Qwen				Avg.
		2-7B	2-13B	3.1-8B	3.2-3B	3-4B	3-8B	3-14B	3-32B	
16-16	FP16	69.49	72.33	74.42	68.20	66.84	70.37	73.82	74.86	71.29
4-4	MXFP4	65.57	69.39	67.61	62.41	61.13	65.80	70.52	73.45	66.98
	NVFP4	68.30	71.41	72.27	66.06	63.28	69.15	72.02	73.62	69.51
	MR-GPTQ	68.69	71.59	72.70	66.18	63.89	68.95	72.45	74.11	69.82
	4over6	68.42	71.34	**73.16**	66.00	63.74	68.71	72.54	74.23	69.77
	RaZeR	**68.89**	**72.12**	72.98	**66.65**	**63.93**	**69.23**	**72.86**	**74.49**	**70.14**

Table 4: Reasoning task accuracy (↑) under different 4-bit weight-only and weight-activation quantization methods.

#Bits W-A	Method	GSM8K		MMLU		Avg.
		3.1-8B	3.2-3B	3.1-8B	3.2-3B	
16-16	FP16	85.52	78.09	72.65	65.25	75.37
4-16	MXFP4	78.16	72.86	68.04	61.10	70.04
	NVFP4	82.11	75.06	70.46	62.04	72.42
	GPTQ	83.24	73.31	70.83	62.57	72.49
	AWQ	78.49	75.74	70.18	62.62	71.76
	RaZeR	**84.01**	**77.26**	**71.34**	**63.10**	**73.92**
4-4	MXFP4	61.49	57.32	57.97	55.68	58.11
	NVFP4	77.71	71.95	68.44	61.00	69.78
	MR-GPTQ	78.43	71.49	68.85	61.29	69.99
	4over6	78.17	72.48	68.99	61.44	70.27
	RaZeR	**82.64**	**75.74**	**69.88**	**62.11**	**72.59**

Zero-shot Accuracy. Table 3 summarizes the average zero-shot task accuracy for weight-activation quantization across various methods. The detailed results for each task can be found in Appendix C.2. RaZeR generally outperforms prior approaches, surpassing the baseline NVFP4 format by 0.63% better accuracy averaged over all models. We note that these zero-shot tasks are generally simple, consisting of primarily multiple-choice questions. As a result, most baseline methods already achieve strong performance, while RaZeR is able to further improve the quantized model accuracy through the novel redundant zero remapping technique.

Reasoning Accuracy. Table 4 summarizes the reasoning task accuracy for Llama-3.1-8B and Llama-3.2-3B, under different weight-only and weight-activation quantization methods. RaZeR demonstrates significant accuracy improvement over prior approaches. For instance, on Llama-3.1-8B running GSM8K, RaZeR outperforms recent NVFP4-based quantization algorithms, MR-GPTQ and FourOverSix, by a large margin of 4.21% and 4.47%, respectively, under weight-activation quantization.

5.3 GPU Kernel Performance

We evaluate the weight-only quantized RaZeR kernel across multiple Blackwell GPU platforms, including the datacenter-class NVIDIA RTX Pro 6000 Server Edition, the desktop edge device DGX Spark, and the RTX 5090. We compare against representative high-performance weight-only 4-bit inference kernels, including AWQ [25], Any-Precision-LLM [37], SqueezeLLM [23], and Marlin [15]. These methods span both uniformly scaled INT4 quantization (e.g., AWQ, Marlin) and lookup-table–based 4-bit quantization (e.g., Any-Precision-LLM, SqueezeLLM). We also evaluate a modified Marlin variant that replaces INT4 weights with FP4 quantization in order to quantify the overhead of the redundant zero remap operation. All 4-bit kernel implementations are integrated into the `gpt-fast` [31] framework to measure end-to-end autoregressive decoding throughput over 200 generated tokens.

Fig. 4 reports the performance results on RTX Pro 6000, RTX 5090, and DGX Spark for two representative models. Across different platforms, the RaZeR kernel achieves near-best-in-class single-batch decoding performance. At moderate to large batch sizes, its throughput is only slightly below Marlin and Marlin-FP4, while substantially outperforming AWQ and other weight-only

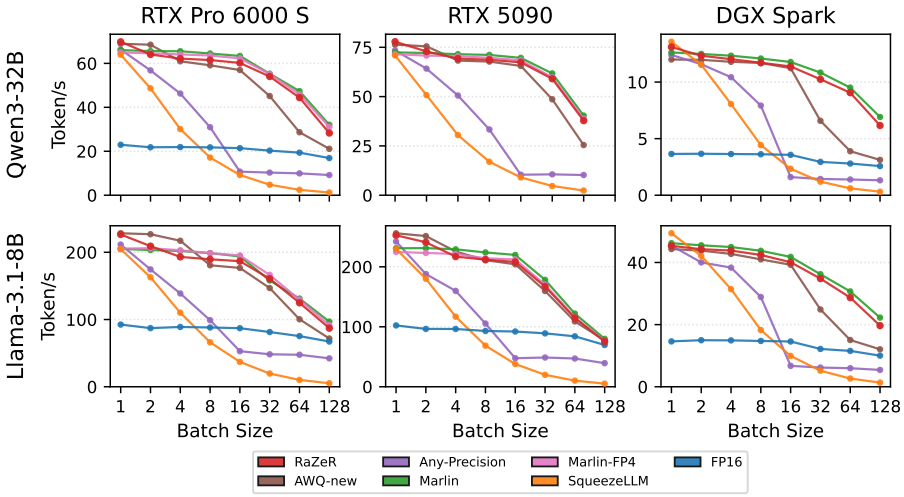


Figure 4: LLM end-to-end decode speed measurements on three devices. Token generation speed is reported across batch sizes. Missing data points correspond to out-of-memory configurations.

Table 5: Area and power of the baseline NVFP4 tensor core and the proposed RaZeR tensor core.

	Area (μm^2)			Power (mW)		
	Array	Decoder	Total	Array	Decoder	Total
NVFP4	2.315 E5	391	2.319 E5	40.8	0.6	41.4
RaZeR	2.393 E5	1201	2.405 E5	45.7	1.3	47.0

quantized inference kernels. Because RaZeR preserves substantially higher accuracy than the plain INT4 and FP4 formats used by Marlin and Marlin-FP4, it lies on the accuracy–throughput Pareto frontier among the evaluated kernels. Additional GPU evaluation results are provided in Appendix D, including results on a larger diversity of models, as well as isolated kernel microbenchmarks.

5.4 Hardware Cost Analysis

To examine the hardware cost of adopting RaZeR on future accelerator platforms, we model the proposed custom tensor core (Fig. 3 right) in SystemVerilog, which is synthesized by Synopsys Design Compiler under TSMC 28nm technology to estimate the area and power. As shown in Table 5, the modified RaZeR tensor core incurs 3.7% / 13.5% area / power overhead on top of the baseline NVFP4 tensor core. Given that MAC units typically occupy less than 10% of the entire design within modern DNN accelerators [22], the relative chip area / power overhead of supporting RaZeR is merely 0.37% / 1.35%.

6 Conclusion

We introduce RaZeR, an enhanced NVFP4 format that exploits redundancy in the block scaling factor to repurpose the redundant FP4 zero as an optimized special value. This approach improves quantization accuracy without incurring additional memory overhead. Complementing this, we implement efficient GPU kernels and a MAC unit to demonstrate the practicality of adopting RaZeR in current and future hardware platforms. Our evaluations show that RaZeR outperforms existing 4-bit methods across both weight-only and weight-activation quantization, while incurring only 0.37% chip area overhead. Compared to the state-of-the-art 4-bit quantization method FourOverSix, RaZeR reduces the average perplexity loss by 29.2% and 23.3% under weight-only and weight-activation quantization, respectively. We hope this work can motivate future research on designing efficient low-precision numerics for LLM quantization.

Limitations: Although RaZeR enables more accurate NVFP4 quantization, work remains to support RaZeR in hardware. In the future, we plan to tape-out a RaZeR tensor core to more accurately measure its area and power overhead. In addition, our work finds that the E3M3 weight scale is

lossless for Llama and Qwen. However, in practice, other LLMs might exhibit different characteristics that prefer the baseline E4M3 scale. While RaZeR can still be employed under such a situation, the number of weight special values will be reduced from 4 to 2.

Acknowledgments and Disclosure of Funding

This project is supported in part by the National Science Foundation under Grant No. 2339084.

References

[1] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019.

[2] Mengzhao Chen, Meng Wu, Hui Jin, Zhihang Yuan, Jing Liu, Chaoyi Zhang, Yunshui Li, Jie Huang, Jin Ma, Zeyue Xue, Zhiheng Liu, Xingyan Bin, and Ping Luo. Int v.s. fp: A comprehensive study of fine-grained low-bit quantization formats. *arXiv preprint arXiv:2510.25602*, 2025.

[3] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[4] 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.

[5] Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, and Song Han. Four Over Six: More Accurate NVFP4 Quantization with Adaptive Block Scaling. *arXiv preprint arXiv:2512.02010*, 2025.

[6] Steve Dai, Rangharajan Venkatesan, Haoxing Ren, Brian Zimmer, William J. Dally, and Brucek Khailany. VS-Quant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems (MLSys)*, 2021.

[7] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems (NeurIPS)*, 2022.

[8] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit Optimizers via Block-wise Quantization. *International Conference on Learning Representations (ICLR)*, 2022.

[9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in neural information processing systems (NeurIPS)*, 2023.

[10] Jesse Dodge, Ana Marasovic, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[11] Jordan Dotzel, Yuzong Chen, Bahaa Kotb, Sushma Prasad, Gang Wu, Sheng Li, Mohamed S. Abdelfattah, and Zhiru Zhang. Learning from Students: Applying t-Distributions to Explore Accurate and Efficient Formats for LLMs. *International Conference on Machine Learning (ICML)*, 2024.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

[13] Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the Gap Between Promise and Performance for Microscaling FP4 Quantization. *International Conference on Learning Representations (ICLR)*, 2026.

[14] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-training Compression for Generative Pretrained Transformers. *International Conference on Learning Representations (ICLR)*, 2023.

[15] Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. *arXiv preprint arXiv:2408.11743*, 2024.

[16] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, 2020.

[17] 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, 2024.

[18] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

[20] Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. OstQuant: Refining Large Language Model Quantization with Orthogonal and Scaling Transformations for Better Distribution Fitting. *International Conference on Machine Learning (ICML)*, 2025.

[21] Wonsuk Jang and Thierry Tambe. BlockDialect: Block-wise Fine-grained Mixed Format Quantization for Energy-Efficient LLM Inference. *International Conference on Machine Learning (ICML)*, 2025.

[22] Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. *ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2023.

[23] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-Sparse Quantization. *International Conference on Machine Learning (ICML)*, 2024.

[24] Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing Outliers via Dual Transformation Makes Stronger Quantized LLMs. *Advances in neural information processing systems (NeurIPS)*, 2024.

[25] 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 LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

[26] 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 (MLSys)*, 2025.

[27] Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. Llm-fp4: 4-bit floating-point quantized transformers. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[28] 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, 2025.

[29] Yun-Chen Lo, Gu-Yeon Wei, and David Brooks. Nanoscaling Floating-Point (NxFP): NanoMantissa, Adaptive Microexponents, and Code Recycling for Direct-Cast Compression of Large Language Models. arXiv preprint arXiv:2412.19821, 2024.

[30] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *International Conference on Learning Representations (ICLR)*, 2017.

[31] meta-pytorch contributors. gpt-fast: Simple and efficient pytorch-native transformer text generation, 2025.

[32] Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep K. Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart F. Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 Formats for Deep Learning. *arXiv preprint arXiv:2209.05433*, 2022.

[33] NVIDIA. Introducing NVFP4 for Efficient and Accurate Low-Precision Inference, 2025.

[34] NVIDIA. NVIDIA Blackwell GPU Architecture, 2025.

[35] Open Compute Project. OCP Microscaling Formats (MX) Specification, 2023.

[36] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics.

[37] Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. Any-precision llm: Low-cost deployment of multiple, different-sized llms. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[38] Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, Lai Shao, Gaurav Kolhe, Dimitry Melts, Jasmine Klar, Renee L'Heureux, Matt Perry, Doug Burger, Eric S. Chung, Zhaoxia Deng, Sam Naghshineh, Jongsoo Park, and Maxim Naumov. With Shared Microexponents, A Little Shifting Goes a Long Way. *ACM/IEEE 50th Annual International Symposium on Computer Architecture (ISCA)*, 2023.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

[40] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqiang Li, Kaipeng Zhang, Peng Gao, Yu Jiao Qiao, and Ping Luo. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. *International Conference on Learning Representations (ICLR)*, 2024.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[42] Xiaoxia Wu, Zhewei Yao, and Yuxiong He. ZeroQuant-FP: A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats. *arXiv preprint arXiv:2307.09782*, 2023.

[43] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. *International Conference on Machine Learning (ICML)*, 2023.

[44] 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*, 2025.

[45] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[46] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit Quantization for Efficient and Accurate LLM Serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Appendix Overview

- Section A: Additional Ablation Study on Block Scale Format.
- Section B: Additional Experimental Details for Accuracy Evaluation.
- Section C: Additional Accuracy Results.
- Section D: Additional Kernel Evaluation.
- Section E: Additional Experiments on Auto-tuned Kernel Performance.

Code availability. Our code, including quantization and evaluation scripts, inference implementations, and setup instructions, is publicly available at `https://github.com/abdelfattah-lab/NVFP4-RaZeR`.

A Additional Ablation Study on Block Scale Format

Table 6 and Table 7 summarize the full perplexity results when applying different block scale formats to NVFP4 weight-only and activation-only quantization, respectively. Weights are generally more tolerant to a scale format with reduced bit width. Activations are more sensitive to reduced exponent bits due to their large dynamic range.

Table 6: Wikitext-2 and C4 perplexity (↓) using different block scale formats for weight-only NVFP4 quantization. Continuation of Table 1. We highlight cases where the average perplexity is the same or better than that of baseline E4M3 scale.

		Llama								Qwen								
#Bits	**Format**	2-7B		2-13B		3.1-8B		3.2-3B		3-4B		3-8B		3-14B		3-32B		**Avg.**
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	
8	E5M3	5.63	7.15	4.98	6.56	6.63	9.48	8.24	11.10	13.63	16.85	9.92	13.55	8.80	12.23	7.87	10.99	**9.60**
	E4M4	5.63	7.15	4.98	6.56	6.63	9.47	8.26	11.11	13.82	16.81	10.09	13.58	8.78	12.22	7.84	10.98	9.62
	E3M5	5.63	7.14	4.98	6.56	6.63	9.47	8.26	11.11	13.67	16.84	9.97	13.53	8.80	12.21	7.85	10.98	**9.60**
7	E5M2	5.65	7.15	4.99	6.58	6.67	9.53	8.29	11.17	13.96	16.89	9.99	13.59	8.82	12.29	7.85	10.97	9.65
	E4M3	5.63	7.15	4.98	6.56	6.63	9.48	8.24	11.10	13.63	16.85	9.92	13.55	8.80	12.23	7.87	10.99	**9.60**
	E3M4	5.63	7.15	4.98	6.56	6.63	9.47	8.26	11.11	13.63	16.81	9.92	13.60	8.78	12.22	7.84	10.97	**9.60**
6	E4M2	5.65	7.15	4.99	6.58	6.67	9.53	8.29	11.17	13.96	16.89	9.99	13.59	8.82	12.29	7.85	10.97	9.65
	E3M3	5.63	7.15	4.97	6.56	6.63	9.48	8.24	11.10	13.63	16.85	9.92	13.55	8.80	12.24	7.87	10.99	**9.60**
	E2M4	5.64	7.16	4.99	6.57	6.64	9.5	8.25	11.11	14.13	16.97	10.11	13.65	8.83	12.29	7.85	10.97	9.67
5	E3M2	5.65	7.15	4.99	6.58	6.67	9.53	8.29	11.17	13.96	16.9	10.00	13.59	8.82	12.3	7.85	10.97	9.65
	E2M3	5.69	7.19	5.02	6.61	6.72	9.63	8.32	11.21	14.22	17.22	10.22	13.79	9.02	12.48	7.93	11.13	9.78

Table 7: Wikitext-2 and C4 perplexity (↓) using different block scale formats for activation-only NVFP4 quantization. Continuation of Table 1. We highlight cases where the average perplexity is the same or better than that of baseline E4M3 scale.

		Llama								Qwen								
#Bits	**Format**	2-7B		2-13B		3.1-8B		3.2-3B		3-4B		3-8B		3-14B		3-32B		**Avg.**
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	
8	E5M3	5.59	7.09	4.97	6.55	6.53	9.35	8.11	10.9	13.85	16.93	9.84	13.5	8.76	12.16	7.74	10.93	**9.55**
	E4M4	5.58	7.08	4.96	6.55	6.51	9.33	8.11	10.87	13.81	16.9	9.83	13.47	8.74	12.15	7.72	10.92	**9.53**
	E3M5	5.61	7.11	4.96	6.55	6.53	9.36	8.12	10.9	13.80	16.89	9.85	13.48	8.73	12.15	7.72	10.93	**9.54**
7	E5M2	5.62	7.12	4.99	6.58	6.58	9.43	8.18	10.98	13.97	17.03	9.91	13.57	8.81	12.20	7.77	10.97	9.61
	E4M3	5.59	7.09	4.97	6.55	6.53	9.35	8.12	10.89	13.81	16.93	9.84	13.49	8.77	12.16	7.74	10.93	**9.55**
	E3M4	5.67	7.18	4.97	6.55	6.61	9.43	8.18	10.97	13.89	16.97	9.89	13.53	8.74	12.16	7.73	10.93	9.59
6	E4M2	5.62	7.12	4.98	6.57	6.59	9.43	8.18	10.98	13.94	17.02	9.92	13.58	8.80	12.21	7.77	10.97	9.61
	E3M3	5.84	7.32	5.0	6.58	6.89	9.78	8.43	11.27	14.01	17.26	10.02	13.71	8.77	12.23	7.71	10.97	9.74
	E2M4	17.47	8.43	5.58	7.37	12.66	20.41	11.83	15.28	14.63	18.16	10.43	14.26	8.99	12.52	7.87	11.33	12.33
5	E3M2	6.75	7.68	5.17	6.75	8.18	11.74	9.50	12.54	14.47	17.88	10.29	14.09	8.91	12.33	7.77	11.08	10.32
	E2M3	15.93	9.59	6.36	8.39	15.77	26.49	13.13	17.51	15.01	18.47	10.48	14.33	9.22	12.81	8.10	11.50	13.32

B Additional Experimental Details for Accuracy Evaluation

B.1 Details for Baseline Methods

We provide additional details about the baseline quantization methods used in our evaluation.

- MXFP4: A variant of the microscaling (MX) format, which is standardized by [35]. A concrete MX-compliant format defines three items: the data format, the block size, and the block scale format. By default, the MX block size is 32 with a block scale format of E8M0. The data format can be INT8, FP8-E5M2, FP8-E4M3, FP6-E3M2, FP6-E2M3, FP4-E2M1.
- NVFP4: A custom FP4 format introduced since Blackwell GPU [33]. It applies a fine-grained FP8-E4M3 scaling factor to each 16-value block, while also leveraging a second-level FP32 scaling factor applied per tensor.
- NF4: The 4-bit NormalFloat format introduced in QLoRA [9]. It leverages the statistical quantile function to derive the 16 quantization values that are information theoretically optimal for normally distributed tensors. The 16 quantization values are stored in a high-precision BF16 lookup table. During computation, every quantized 4-bit element serves as an index to extract the corresponding BF16 value from the lookup table.
- BlockDialect [21]: A 4-bit block-wise mixed format that enhances MXFP4. It assigns a per-block optimal number format from a formatbook to achieve better data representation. To preserve compatibility with energy-efficient low-precision arithmetic, it introduces DialectFP4, a formatbook of FP4 variants that adapt to diverse data distributions.
- GPTQ [14]: A weight-only quantization method that uses the second-order Hessian information to perform error compensation.
- AWQ [25]: A weight-only quantization method that protects the salient weight channels corresponding to larger activation magnitudes. It performs adaptive scaling to scale up the salient weight channels, which can reduce their relative quantization error. In addition, it applies weight clipping to further minimize the quantization error.
- SqueezeLLM [23]: A weight-only quantization method that (1) leverages sensitivity-based non-uniform quantization to search for the optimal bit precision assignment based on second-order information, and (2) employs dense-and-sparse decomposition that stores outlier weights in FP16 using an efficient sparse format.
- MR-GPTQ [13]: A weight-activation quantization method that applies Hadamard transformation to activations, while using GPTQ to reduce the quantization error of weights.
- FourOverSix [5]: A modification to the NVFP4 quantization algorithm that evaluates two potential scale factors for each block of values, one by scaling to the full range of FP4 values $-6 \sim 6$, and another by scaling to a narrower range of $-4 \sim 4$. It then selects the optimal scaling based on the mean-squared quantization error.

B.2 Quantization Details for RaZeR

In our evaluation, both weights and activations use ± 5 in their set of allowed special values. The weights allow two extra special values, which vary across different models as summarized in Table 8.

Table 8: Four special values for RaZeR weights in different models.

Llama				**Qwen**			
2-7B	2-13B	3.1-8B	3.2-3B	3-4B	3-8B	3-14B	3-32B
$\pm 5, \pm 8$	$\pm 5, \pm 8$	$\pm 5, \pm 8$	$\pm 5, \pm 8$	$\pm 5, \pm 8$	$\pm 5, \pm 7$	$\pm 5, \pm 8$	$\pm 5, \pm 9$

C Additional Accuracy Results

C.1 Perplexity under Joint Quantization of Weights, Activations, and KV-cache

We compare the perplexity of different 4-bit quantization methods under joint quantization of weights, activations, and KV-cache. Specifically, we compare with MXFP4 [35], NVFP4 [33], NF4 [9], Atom [46], OmniQuant [40], DuQuant [24], SpinQuant [28], OSTQuant [20], and FourOverSix [5]. Since some baseline methods do not support Qwen, we focus on evaluating Llama models as shown in Table 9. RaZeR delivers better performance across all evaluated models and datasets.

Table 9: Perplexity (↓) under joint quantization of weights, activations, and KV-cache, using different 4-bit quantization methods.

Method	**Llama-1**						**Llama-2**						**Llama-3**			
	1-7B		1-13B		1-65B		2-7B		2-13B		2-70B		3.1-8B		3.2-3B	
	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
FP16	5.68	7.08	5.09	6.61	3.53	5.62	5.47	6.97	4.88	6.46	3.31	5.52	6.24	8.96	7.82	10.44
MXFP4	7.91	9.75	6.39	8.01	4.80	6.89	8.33	11.39	7.06	10.32	4.60	6.68	10.26	13.96	12.50	16.14
NVFP4	6.06	7.44	5.33	6.84	3.78	5.77	5.85	7.38	5.13	6.73	3.54	5.68	7.14	10.19	8.93	12.08
NF4	6.04	7.51	5.40	6.89	3.79	5.81	5.93	7.48	5.21	6.77	3.60	5.71	7.40	10.54	9.35	12.54
Atom	6.16	7.70	5.46	7.04	3.89	5.92	6.04	7.75	5.27	6.95	3.68	5.83	–	–	–	–
OmniQuant	11.26	14.51	10.87	13.78	9.17	11.28	14.26	18.02	12.30	14.55	–	–	–	–	–	–
DuQuant	6.18	7.73	5.47	7.07	3.93	5.93	6.08	7.79	5.33	7.02	3.76	5.85	–	–	–	–
SpinQuant	6.12	–	5.39	–	–	–	5.96	–	5.24	–	3.70	–	–	–	–	–
OSTQuant	6.07	–	5.40	–	–	–	5.91	–	5.25	–	3.59	–	–	–	–	–
4over6	6.04	7.42	5.30	6.81	3.75	5.75	5.79	7.32	5.11	6.69	3.51	5.66	7.03	10.02	8.79	11.87
RaZeR	**5.93**	**7.32**	**5.24**	**6.76**	**3.71**	**5.72**	**5.72**	**7.25**	**5.06**	**6.64**	**3.47**	**5.63**	**6.86**	**9.78**	**8.56**	**11.53**

C.2 Detailed Zero-shot Task Accuracy

We provide the detailed zero-shot task accuracy for Llama and Qwen3 models in Table 10 and Table 11, respectively.

Table 10: Detailed zero-shot task accuracy (↑) for Llama models under 4-bit weight-activation quantization. KV-cache remains in FP16.

Model	Method	PIQA	Hellaswag	Winogrande	ARC-e	ARC-c	Lambada	Avg.
Llama-2-7B	FP16	78.45	76.17	70.24	73.82	44.97	73.28	69.49
	MXFP4	76.61	71.31	65.27	69.15	41.38	69.69	65.57
	NVFP4	77.91	75.07	68.98	71.89	43.17	72.77	68.30
	MR-GPTQ	77.86	74.99	68.82	72.94	44.37	73.14	68.69
	4over6	78.35	75.15	69.14	72.43	43.05	72.40	68.42
	RaZeR	78.18	75.16	69.82	72.69	44.49	72.97	**68.89**
Llama-2-13B	FP16	80.63	79.64	72.22	76.52	48.63	76.36	72.33
	MXFP4	77.97	74.05	71.19	73.99	45.22	73.90	69.39
	NVFP4	79.92	78.54	71.19	75.51	47.27	76.05	71.41
	MR-GPTQ	79.43	78.84	71.27	75.67	47.82	76.50	71.59
	4over6	79.98	78.84	71.43	75.00	46.93	75.88	71.34
	RaZeR	80.20	78.91	73.40	75.80	48.38	76.03	**72.12**
Llama-3.1-8B	FP16	80.96	79.41	74.11	82.41	54.95	74.66	74.42
	MXFP4	76.01	73.75	68.90	73.40	46.42	67.16	67.61
	NVFP4	78.67	77.98	72.22	78.54	52.56	73.63	72.27
	MR-GPTQ	79.43	77.23	71.74	80.64	52.90	74.23	72.70
	4over6	79.98	78.17	72.22	80.01	54.61	73.98	**73.16**
	RaZeR	80.14	77.94	72.38	80.13	53.16	74.13	72.98
Llama-3.2-3B	FP16	77.69	74.08	69.53	72.05	46.16	69.71	68.20
	MXFP4	74.59	67.76	64.96	66.04	41.04	60.08	62.41
	NVFP4	75.63	71.61	67.40	69.82	44.45	67.46	66.06
	MR-GPTQ	76.93	71.68	66.54	70.16	44.80	66.99	66.18
	4over6	75.52	71.91	67.64	69.99	43.60	67.34	66.00
	RaZeR	76.39	72.43	70.01	69.95	43.00	68.14	**66.65**

Table 11: Detailed zero-shot task accuracy (↑) for Qwen3 models under 4-bit weight-activation quantization. KV-cache remains in FP16.

Model	Method	PIQA	Hellaswag	Winogrande	ARC-e	ARC-c	Lambada	Avg.
Qwen3-4B	FP16	75.03	68.54	65.67	78.32	54.10	59.36	66.84
	MXFP4	72.80	63.40	62.27	67.89	46.84	53.56	61.13
	NVFP4	73.88	66.31	61.56	73.36	50.68	53.87	63.28
	MR-GPTQ	73.39	66.11	62.59	75.13	49.91	56.18	63.89
	4over6	72.80	66.61	62.83	74.37	50.26	55.54	63.74
	RaZeR	73.07	66.52	63.69	74.37	50.94	54.96	**63.93**
Qwen3-8B	FP16	77.75	74.96	67.88	80.98	56.40	64.22	70.37
	MXFP4	73.45	69.31	64.48	75.17	52.47	59.93	65.80
	NVFP4	76.22	73.29	67.96	79.88	54.95	62.62	69.15
	MR-GPTQ	76.66	72.75	67.80	79.88	53.84	62.76	68.95
	4over6	75.63	73.25	67.64	78.79	54.01	62.91	68.71
	RaZeR	76.55	73.68	68.19	80.56	53.84	62.57	**69.23**
Qwen3-14B	FP16	79.92	78.80	72.93	82.70	60.67	67.92	73.82
	MXFP4	77.86	75.51	69.53	79.92	56.57	63.75	70.52
	NVFP4	78.51	77.93	72.22	80.72	57.59	65.13	72.02
	MR-GPTQ	79.50	77.63	71.51	80.85	58.36	66.74	72.45
	4over6	78.29	78.52	71.59	81.48	60.24	65.11	72.54
	RaZeR	79.60	78.45	71.82	81.61	59.90	65.77	**72.86**
Qwen3-32B	FP16	81.94	82.59	73.40	83.25	60.84	67.13	74.86
	MXFP4	80.14	80.43	71.35	81.99	60.41	66.39	73.45
	NVFP4	79.87	81.93	69.85	82.03	61.18	66.85	73.62
	MR-GPTQ	81.66	81.90	71.51	82.53	59.90	67.18	74.11
	4over6	80.79	81.97	72.06	82.74	60.67	67.16	74.23
	RaZeR	80.74	82.17	73.09	82.58	61.60	66.78	**74.49**

C.3 Ablation Studies

We apply RaZeR to weights and activations independently to quantify its benefits. As shown in Table 12, RaZeR improves the accuracy of both activation-only and weight-only quantization compared to the baseline NVFP4 format. The lowest perplexity is achieved when both weights and activations adopt RaZeR. Notably, applying RaZeR only to activations achieves performance comparable to the state-of-the-art method FourOverSix, while applying it only to weights surpasses FourOverSix. These results highlight the effectiveness of RaZeR for enhanced NVFP4 quantization.

Table 12: Ablation study of RaZeR applied to activation-only, weight-only, and weight-activation quantization for Llama models.

Format	**2-7B**		**2-13B**		**3.1-8B**		**3.2-3B**	
W-A	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
NVFP4-NVFP4	5.77	7.29	5.07	6.66	6.95	9.94	8.60	11.65
4over6-4over6	5.72	7.25	5.06	6.63	6.88	9.83	8.53	11.53
NVFP4-RaZeR	5.73	7.25	5.05	6.63	6.87	9.82	8.53	11.52
RaZeR-NVFP4	5.69	7.22	5.05	6.62	6.80	9.73	8.46	11.38
RaZeR-RaZeR	**5.66**	**7.19**	**5.01**	**6.59**	**6.74**	**9.63**	**8.37**	**11.27**

C.4 Impact of Quantization Block Size

While the original NVFP4 format has a fixed block size of 16 and an effective bit-width of 4.5, one can flexibly adjust the block size to reduce the memory and computational overhead of block scaling. We explore the impact of block size on NVFP4, FourOverSix, and RaZeR, using different Llama models. As shown in Table 13, RaZeR consistently achieves the best perplexity across different block sizes, thanks to its additional quantization value. In particular, the advantage of FourOverSix over NVFP4 gradually diminishes as the block size increases. This is because FourOverSix allows each block to be scaled either to the full FP4 range of $-6 \sim 6$, or to a narrower range of $-4 \sim 4$. At small block sizes, scaling to $-4 \sim 4$ reduces the quantization error of near-maximal values, thus improving model performance. However, with larger block sizes, scaling to $-4 \sim 4$ discards two available FP4 values and is therefore rarely adopted by any block, resulting in performance similar to NVFP4.

Table 13: Impact of block size on perplexity (↓), under different 4-bit weight-activation quantization methods for Llama.

Size	**Method**	**2-7B**		**2-13B**		**3.1-8B**		**3.2-3B**	
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
16	NVFP4	5.77	7.29	5.07	6.66	6.95	9.94	8.60	11.65
	4over6	5.72	7.25	5.06	6.63	6.88	9.83	8.53	11.53
	RaZeR	**5.66**	**7.19**	**5.01**	**6.59**	**6.74**	**9.63**	**8.37**	**11.27**
32	NVFP4	5.83	7.37	5.12	6.71	7.10	10.17	8.82	11.93
	4over6	5.81	7.35	5.11	6.69	7.07	10.11	8.77	11.84
	RaZeR	**5.74**	**7.27**	**5.06**	**6.65**	**6.89**	**9.88**	**8.56**	**11.54**
64	NVFP4	5.91	7.47	5.20	6.78	7.32	10.45	9.08	12.24
	4over6	5.90	7.46	5.20	6.77	7.30	10.40	9.06	12.21
	RaZeR	**5.82**	**7.37**	**5.13**	**6.71**	**7.09**	**10.15**	**8.80**	**11.90**
128	NVFP4	6.05	7.62	5.31	6.87	7.61	10.76	9.48	12.73
	4over6	6.06	7.62	5.29	6.86	7.62	10.78	9.51	12.72
	RaZeR	**5.96**	**7.52**	**5.21**	**6.80**	**7.39**	**10.50**	**9.14**	**12.35**

C.5 Combination with Other Quantization Approaches

We combine RaZeR with AWQ [25], a state-of-the-art weight-only quantization scheme, to demonstrate its potential synergy with existing approaches. Table 14 reports the perplexity of various Llama models under 4-bit AWQ using a block size of 128 and different weight formats. Incorporating RaZeR significantly reduces model perplexity compared to the baseline AWQ implementation using INT4 weight format. These results highlight the robustness of RaZeR under established quantization optimizations.

Table 14: Llama model perplexity (↓) of AWQ weight-only quantization combined with different formats.

Method	2-7B		2-13B		3.1-8B		3.2-3B	
	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
AWQ + INT4	5.68	7.19	5.01	6.60	6.80	9.72	8.50	11.38
AWQ + FP4	5.63	7.14	4.99	6.58	6.68	9.56	8.32	11.18
AWQ + RaZeR	**5.60**	**7.10**	**4.98**	**6.56**	**6.61**	**9.41**	**8.20**	**11.01**

D Additional Kernel Evaluation

D.1 Results across Additional Models and Hardware

We extend the evaluation in the main text to additional models from the Qwen and Llama families with different model sizes and projection shapes, which are omitted from the main text for space. Fig. 5 summarizes end-to-end decoding throughput across eight such models over a range of batch sizes. The observed trends are consistent with the main results: RaZeR achieves near-best-in-class performance at batch size one and maintains competitive throughput at moderate to large batch sizes, closely tracking Marlin and Marlin-FP4 while substantially outperforming AWQ and other weight-only quantized kernels. Results are consistent across the range of evaluated GPUs and models, indicating that the performance characteristics of RaZeR are consistent across different classes of Blackwell devices. Some data points are omitted due to out-of-memory (OOM) conditions for certain model and batch-size combinations on specific devices.

D.2 Weight-Only Kernel Microbenchmarks

To isolate kernel-level performance from end-to-end model effects, we evaluate weight-only matrix multiplication microbenchmarks across a broad set of matrix shapes and batch sizes. Tables 15, 16, and 17 report kernel execution time for representative (M, N, K) configurations corresponding to the linear projections in Llama 3.1 8B and Qwen3-32B on three GPU platforms.

Across all evaluated configurations, RaZeR achieves performance comparable to or exceeding prior state-of-the-art weight-only quantization kernels. We report results for both CUDA-core- and tensor-core-based RaZeR implementations, which exhibit complementary performance regimes depending on batch size. For each configuration, the faster of the two RaZeR variants is shown in red to indicate the best-performing implementation. Overall, the additional metadata access and remapping logic introduce minimal kernel-level overhead, with RaZeR consistently matching or outperforming existing FP4 and INT4 kernels across models, layers, batch sizes, and GPUs.

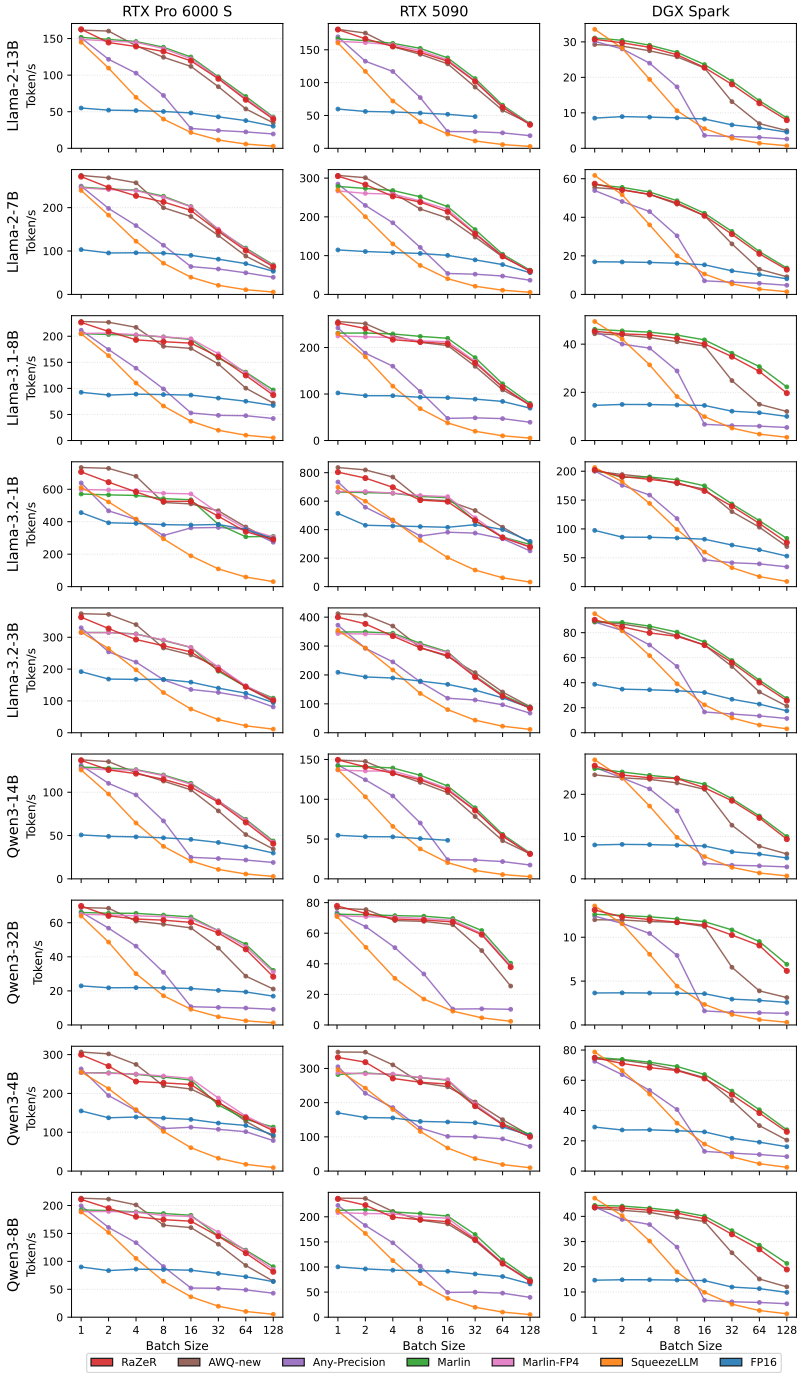


Figure 5: LLM end-to-end decode speed measurements on three GPUs. Token generation speed is reported across batch sizes. Missing data points correspond to out-of-memory configurations.

Table 15: Kernel latency microbenchmark (μs, speedup over FP16) (RTX PRO 6000 S)

Model	Layer	K	N	M	FP16	RaZeR-CUDA	RaZeR-TC	Marlin	Marlin-FP4	Any-Precision	SqueezeLLM	AWQ
Llama-3.1-8B	attn.qkv	4096	6144	1	43.1	12.2 (3.54×)	18.5 (2.33×)	16.9 (2.56×)	15.0 (2.88×)	14.7 (2.94×)	15.7 (2.75×)	19.1 (2.26×)
				2	39.6	16.2 (2.45×)	18.8 (2.11×)	17.2 (2.31×)	15.2 (2.61×)	20.2 (1.96×)	23.0 (1.72×)	19.3 (2.05×)
				4	44.3	20.5 (2.16×)	18.5 (2.39×)	16.9 (2.63×)	14.9 (2.96×)	26.7 (1.66×)	39.5 (1.12×)	19.5 (2.27×)
				8	46.8	37.2 (1.26×)	18.7 (2.51×)	17.1 (2.75×)	15.1 (3.10×)	58.1 (0.81×)	71.6 (0.65×)	19.4 (2.42×)
				16	48.0	68.2 (0.70×)	18.8 (2.56×)	17.6 (2.72×)	15.3 (3.14×)	71.4 (0.67×)	131.9 (0.36×)	20.6 (2.32×)
				32	39.5	129.2 (0.31×)	25.4 (1.55×)	25.6 (1.54×)	22.4 (1.76×)	62.4 (0.63×)	255.2 (0.15×)	33.7 (1.17×)
				64	43.6	248.4 (0.18×)	36.5 (1.20×)	34.6 (1.26×)	33.4 (1.31×)	64.7 (0.67×)	504.5 (0.09×)	49.0 (0.89×)
				128	43.1	445.5 (0.10×)	43.2 (1.00×)	39.8 (1.08×)	39.5 (1.09×)	65.6 (0.66×)	988.8 (0.04×)	74.8 (0.58×)
	attn.o	4096	4096	1	28.3	9.3 (3.03×)	15.5 (1.83×)	16.5 (1.72×)	13.1 (2.17×)	11.9 (2.37×)	11.9 (2.39×)	18.6 (1.52×)
				2	31.5	12.7 (2.47×)	15.3 (2.06×)	16.3 (1.94×)	13.1 (2.40×)	19.0 (1.66×)	16.9 (1.87×)	18.6 (1.69×)
				4	27.3	15.6 (1.75×)	15.7 (1.74×)	16.5 (1.66×)	13.2 (2.07×)	24.6 (1.11×)	28.4 (0.96×)	18.6 (1.47×)
				8	27.5	25.0 (1.10×)	16.1 (1.71×)	16.7 (1.65×)	13.3 (2.06×)	33.9 (0.81×)	49.8 (0.55×)	19.2 (1.43×)
				16	27.3	43.5 (0.63×)	16.4 (1.66×)	17.4 (1.57×)	13.5 (2.03×)	43.8 (0.62×)	93.5 (0.29×)	20.1 (1.36×)
				32	29.4	81.1 (0.36×)	24.8 (1.19×)	29.5 (1.00×)	21.4 (1.37×)	46.1 (0.64×)	175.5 (0.17×)	20.9 (1.41×)
				64	28.2	153.6 (0.18×)	33.7 (0.84×)	38.9 (0.72×)	31.8 (0.89×)	44.6 (0.63×)	341.2 (0.08×)	30.5 (0.92×)
				128	30.6	298.0 (0.10×)	37.0 (0.83×)	35.5 (0.86×)	33.6 (0.91×)	47.1 (0.65×)	679.2 (0.05×)	38.0 (0.81×)
	mlp.gateup	4096	28672	1	166.5	44.3 (3.76×)	49.4 (3.37×)	47.5 (3.51×)	46.9 (3.55×)	50.5 (3.30×)	51.8 (3.22×)	58.3 (2.85×)
				2	163.2	48.9 (3.34×)	51.6 (3.16×)	46.9 (3.48×)	46.0 (3.54×)	57.4 (2.84×)	84.2 (1.94×)	58.7 (2.78×)
				4	163.6	76.5 (2.14×)	50.6 (3.24×)	48.0 (3.41×)	46.8 (3.49×)	95.0 (1.72×)	145.7 (1.12×)	58.5 (2.80×)
				8	164.5	144.0 (1.14×)	50.0 (3.29×)	47.1 (3.49×)	47.3 (3.48×)	163.4 (1.01×)	268.6 (0.61×)	59.1 (2.78×)
				16	165.1	275.8 (0.60×)	56.8 (2.91×)	48.4 (3.41×)	49.1 (3.36×)	335.3 (0.49×)	515.7 (0.32×)	62.1 (2.66×)
				32	179.4	521.5 (0.34×)	64.6 (2.77×)	52.1 (3.44×)	56.0 (3.20×)	352.8 (0.51×)	1011.2 (0.18×)	90.8 (1.97×)
				64	171.8	994.1 (0.17×)	74.3 (2.31×)	68.2 (2.52×)	71.0 (2.42×)	341.5 (0.50×)	1995.7 (0.09×)	126.2 (1.36×)
				128	183.5	1936.2 (0.09×)	130.6 (1.41×)	118.0 (1.55×)	118.0 (1.55×)	353.7 (0.52×)	3973.5 (0.05×)	192.4 (0.95×)
	mlp.down	14336	4096	1	84.9	23.6 (3.59×)	32.6 (2.60×)	32.2 (2.64×)	30.9 (2.74×)	27.1 (3.13×)	28.2 (3.01×)	48.2 (1.76×)
				2	84.8	27.6 (3.07×)	32.4 (2.62×)	32.4 (2.62×)	31.0 (2.73×)	46.4 (1.83×)	45.7 (1.85×)	48.0 (1.77×)
				4	86.0	41.4 (2.07×)	32.6 (2.64×)	32.1 (2.68×)	31.2 (2.75×)	54.3 (1.58×)	79.6 (1.08×)	48.1 (1.79×)
				8	85.1	75.5 (1.13×)	32.8 (2.59×)	33.0 (2.58×)	31.3 (2.72×)	70.0 (1.21×)	143.6 (0.59×)	48.4 (1.76×)
				16	84.3	137.3 (0.61×)	33.6 (2.51×)	33.5 (2.51×)	31.6 (2.67×)	155.3 (0.54×)	271.0 (0.31×)	49.4 (1.70×)
				32	89.0	261.9 (0.34×)	46.3 (1.92×)	46.5 (1.91×)	41.9 (2.12×)	158.5 (0.56×)	529.9 (0.17×)	51.8 (1.72×)
				64	102.8	515.7 (0.20×)	64.6 (1.59×)	60.9 (1.69×)	59.1 (1.74×)	170.1 (0.60×)	1046.0 (0.10×)	96.6 (1.06×)
				128	95.3	1020.6 (0.09×)	82.6 (1.15×)	74.7 (1.28×)	74.3 (1.28×)	165.1 (0.58×)	2065.4 (0.05×)	118.4 (0.81×)
Qwen-3-32B	attn.qkv	5120	10240	1	75.6	22.6 (3.35×)	25.4 (2.98×)	24.5 (3.08×)	23.0 (3.29×)	31.5 (2.40×)	26.9 (2.81×)	26.9 (2.81×)
				2	80.1	26.2 (3.05×)	25.2 (3.17×)	24.9 (3.22×)	22.9 (3.49×)	28.0 (2.86×)	41.2 (1.95×)	27.5 (2.92×)
				4	78.9	37.0 (2.13×)	26.8 (2.94×)	25.1 (3.15×)	23.5 (3.36×)	55.6 (1.42×)	72.1 (1.10×)	27.8 (2.84×)
				8	78.8	68.6 (1.15×)	27.0 (2.92×)	24.7 (3.19×)	23.9 (3.30×)	76.2 (1.03×)	130.8 (0.60×)	28.1 (2.81×)
				16	79.4	124.3 (0.64×)	28.5 (2.78×)	25.1 (3.17×)	24.5 (3.24×)	139.4 (0.57×)	245.8 (0.32×)	28.9 (2.75×)
				32	86.2	237.7 (0.36×)	34.4 (2.50×)	32.1 (2.69×)	32.4 (2.66×)	141.7 (0.61×)	477.4 (0.18×)	43.1 (2.00×)
				64	82.1	464.0 (0.18×)	45.1 (1.82×)	42.3 (1.94×)	41.3 (1.99×)	137.9 (0.60×)	940.5 (0.09×)	63.7 (1.29×)
				128	96.0	896.2 (0.11×)	69.0 (1.39×)	61.7 (1.56×)	62.0 (1.55×)	150.8 (0.64×)	1865.2 (0.05×)	95.0 (1.01×)
	attn.o	8192	5120	1	60.7	18.1 (3.36×)	24.5 (2.47×)	23.4 (2.59×)	21.3 (2.85×)	20.3 (2.99×)	22.3 (2.72×)	30.7 (1.98×)
				2	59.9	22.4 (2.67×)	24.1 (2.49×)	23.7 (2.53×)	20.9 (2.87×)	29.4 (2.04×)	35.8 (1.67×)	30.5 (1.96×)
				4	60.6	29.9 (2.02×)	24.0 (2.52×)	23.9 (2.53×)	20.7 (2.93×)	35.6 (1.70×)	61.7 (0.98×)	30.6 (1.98×)
				8	61.2	56.4 (1.09×)	24.6 (2.49×)	24.3 (2.52×)	21.0 (2.92×)	48.0 (1.28×)	106.2 (0.58×)	31.3 (1.96×)
				16	62.5	102.1 (0.61×)	26.1 (2.39×)	24.7 (2.53×)	21.7 (2.88×)	101.6 (0.61×)	197.0 (0.32×)	32.1 (1.94×)
				32	62.1	192.1 (0.32×)	36.4 (1.71×)	37.1 (1.67×)	32.9 (1.89×)	102.3 (0.61×)	380.8 (0.16×)	33.2 (1.87×)
				64	76.6	374.0 (0.20×)	48.1 (1.59×)	48.1 (1.59×)	45.2 (1.70×)	109.6 (0.70×)	746.2 (0.10×)	58.5 (1.31×)
				128	80.2	735.6 (0.11×)	62.0 (1.29×)	56.5 (1.42×)	55.8 (1.44×)	121.5 (0.66×)	1475.4 (0.05×)	73.5 (1.09×)
	mlp.gateup	5120	51200	1	389.6	96.2 (4.05×)	101.3 (3.85×)	99.6 (3.91×)	100.3 (3.89×)	99.8 (3.90×)	108.5 (3.59×)	115.2 (3.38×)
				2	362.5	101.5 (3.57×)	102.3 (3.54×)	100.1 (3.62×)	100.8 (3.60×)	120.5 (3.01×)	175.0 (2.07×)	116.6 (3.11×)
				4	363.6	161.1 (2.26×)	103.4 (3.52×)	99.9 (3.64×)	99.8 (3.64×)	172.4 (2.11×)	307.4 (1.18×)	116.4 (3.12×)
				8	367.0	318.5 (1.15×)	102.9 (3.56×)	100.5 (3.65×)	100.4 (3.65×)	312.7 (1.17×)	578.8 (0.63×)	117.1 (3.13×)
				16	369.7	598.4 (0.62×)	105.0 (3.52×)	100.6 (3.68×)	101.2 (3.65×)	797.1 (0.46×)	1118.7 (0.33×)	120.4 (3.07×)
				32	395.0	1116.4 (0.35×)	106.6 (3.71×)	102.8 (3.84×)	107.7 (3.67×)	822.1 (0.48×)	2198.5 (0.18×)	193.3 (2.04×)
				64	393.6	2170.1 (0.18×)	144.7 (2.72×)	130.5 (3.02×)	130.4 (3.02×)	824.2 (0.48×)	4360.6 (0.09×)	285.9 (1.38×)
				128	418.0	4277.7 (0.10×)	262.1 (1.59×)	234.7 (1.78×)	233.7 (1.79×)	845.8 (0.49×)	8683.3 (0.05×)	404.7 (1.03×)
	mlp.down	25600	5120	1	191.5	48.4 (3.96×)	54.4 (3.52×)	55.3 (3.47×)	52.4 (3.66×)	51.5 (3.72×)	56.8 (3.37×)	81.6 (2.35×)
				2	187.0	56.2 (3.33×)	54.9 (3.41×)	54.0 (3.46×)	52.4 (3.57×)	76.6 (2.44×)	92.8 (2.02×)	81.4 (2.30×)
				4	187.3	84.5 (2.22×)	55.5 (3.38×)	56.5 (3.31×)	52.9 (3.54×)	88.1 (2.13×)	160.7 (1.17×)	81.7 (2.29×)
				8	187.2	157.8 (1.19×)	56.7 (3.30×)	55.2 (3.39×)	53.1 (3.52×)	110.8 (1.69×)	296.7 (0.63×)	81.7 (2.29×)
				16	189.2	296.3 (0.64×)	58.5 (3.24×)	56.8 (3.33×)	54.8 (3.45×)	393.8 (0.48×)	569.9 (0.33×)	84.6 (2.24×)
				32	192.5	578.5 (0.33×)	76.8 (2.51×)	69.7 (2.76×)	74.3 (2.59×)	396.3 (0.49×)	1115.0 (0.17×)	88.2 (2.18×)
				64	211.3	1148.8 (0.18×)	98.3 (2.15×)	90.2 (2.34×)	89.6 (2.36×)	413.1 (0.51×)	2205.2 (0.10×)	170.0 (1.24×)
				128	229.0	2277.8 (0.10×)	148.0 (1.55×)	133.5 (1.71×)	132.9 (1.72×)	434.6 (0.53×)	4380.7 (0.05×)	208.5 (1.10×)

Table 16: Kernel latency microbenchmark (μs, speedup over FP16) (RTX 5090)

Model	Layer	K	N	M	FP16	RaZeR-CUDA	RaZeR-TC	Marlin	Marlin-FP4	Any-Precision	SqueezeLLM	AWQ
Llama-3.1-8B	attn.qkv	4096	6144	1	41.1	12.6 (3.27×)	18.3 (2.25×)	17.2 (2.40×)	15.9 (2.58×)	14.2 (2.89×)	15.5 (2.66×)	17.8 (2.31×)
				2	40.9	17.5 (2.34×)	18.1 (2.25×)	17.3 (2.36×)	15.9 (2.57×)	18.6 (2.20×)	23.3 (1.75×)	17.8 (2.30×)
				4	41.2	20.6 (2.00×)	18.4 (2.24×)	17.1 (2.41×)	16.0 (2.58×)	25.2 (1.64×)	39.6 (1.04×)	18.0 (2.29×)
				8	42.4	36.0 (1.18×)	18.3 (2.31×)	17.8 (2.38×)	15.9 (2.66×)	55.6 (0.76×)	71.6 (0.59×)	18.0 (2.36×)
				16	46.9	67.0 (0.70×)	18.1 (2.59×)	18.0 (2.60×)	16.5 (2.84×)	67.7 (0.69×)	135.2 (0.35×)	19.0 (2.47×)
				32	43.7	127.9 (0.34×)	29.4 (1.49×)	27.8 (1.57×)	27.4 (1.60×)	67.4 (0.65×)	259.9 (0.17×)	30.7 (1.43×)
				64	44.2	248.9 (0.18×)	43.5 (1.02×)	42.1 (1.05×)	42.4 (1.04×)	67.7 (0.65×)	510.7 (0.09×)	44.8 (0.99×)
				128	73.3	453.1 (0.16×)	59.2 (1.24×)	56.2 (1.30×)	56.6 (1.29×)	94.9 (0.77×)	994.8 (0.07×)	65.3 (1.12×)
	attn.o	4096	4096	1	30.0	9.5 (3.15×)	15.3 (1.96×)	15.3 (1.97×)	14.1 (2.12×)	11.6 (2.59×)	11.8 (2.54×)	17.1 (1.76×)
				2	28.9	12.3 (2.36×)	15.2 (1.90×)	15.3 (1.88×)	14.1 (2.04×)	16.6 (1.73×)	17.6 (1.64×)	17.1 (1.69×)
				4	29.4	15.7 (1.87×)	15.4 (1.91×)	15.1 (1.95×)	14.0 (2.10×)	23.3 (1.26×)	29.5 (1.00×)	17.2 (1.71×)
				8	29.8	25.3 (1.17×)	15.4 (1.93×)	15.7 (1.89×)	14.1 (2.11×)	32.4 (0.92×)	51.8 (0.57×)	17.2 (1.73×)
				16	32.2	44.9 (0.72×)	15.6 (2.07×)	16.0 (2.01×)	14.5 (2.23×)	48.2 (0.67×)	96.4 (0.33×)	18.4 (1.75×)
				32	32.0	85.2 (0.38×)	26.5 (1.21×)	25.9 (1.23×)	25.9 (1.23×)	48.5 (0.66×)	184.9 (0.17×)	18.8 (1.70×)
				64	31.6	163.5 (0.19×)	38.9 (0.81×)	38.8 (0.82×)	38.6 (0.82×)	48.7 (0.65×)	369.2 (0.09×)	26.7 (1.18×)
				128	39.6	315.7 (0.13×)	47.8 (0.83×)	46.5 (0.85×)	47.5 (0.83×)	56.0 (0.71×)	712.8 (0.06×)	33.9 (1.17×)
	mlp.gateup	4096	28672	1	179.7	47.9 (3.75×)	53.4 (3.37×)	53.2 (3.38×)	52.8 (3.41×)	51.1 (3.52×)	51.1 (3.52×)	57.0 (3.15×)
				2	186.3	51.6 (3.61×)	54.1 (3.44×)	52.9 (3.52×)	52.5 (3.55×)	66.4 (2.81×)	84.2 (2.21×)	56.7 (3.28×)
				4	186.6	77.1 (2.42×)	55.1 (3.38×)	52.3 (3.57×)	52.7 (3.54×)	88.8 (2.10×)	149.2 (1.25×)	56.8 (3.28×)
				8	187.0	148.0 (1.26×)	53.7 (3.48×)	54.0 (3.46×)	52.9 (3.54×)	170.0 (1.10×)	271.5 (0.69×)	56.8 (3.29×)
				16	190.6	280.0 (0.68×)	54.1 (3.52×)	54.0 (3.53×)	52.4 (3.64×)	392.5 (0.49×)	520.2 (0.37×)	59.4 (3.21×)
				32	189.3	528.1 (0.36×)	64.4 (2.94×)	61.9 (3.06×)	59.3 (3.19×)	388.6 (0.49×)	1021.7 (0.19×)	98.3 (1.93×)
				64	192.6	1009.7 (0.19×)	109.3 (1.76×)	106.2 (1.81×)	106.5 (1.81×)	394.8 (0.49×)	2024.5 (0.10×)	141.3 (1.36×)
				128	222.5	1979.6 (0.11×)	205.7 (1.08×)	197.2 (1.13×)	199.5 (1.12×)	423.7 (0.53×)	4022.9 (0.06×)	205.9 (1.08×)
	mlp.down	14336	4096	1	99.1	29.1 (3.41×)	31.9 (3.11×)	32.8 (3.02×)	30.5 (3.25×)	29.1 (3.40×)	28.7 (3.45×)	43.5 (2.28×)
				2	93.8	28.9 (3.25×)	31.8 (2.95×)	32.2 (2.91×)	29.8 (3.15×)	41.7 (2.25×)	44.1 (2.13×)	43.4 (2.16×)
				4	93.9	41.0 (2.29×)	32.1 (2.93×)	32.6 (2.88×)	30.2 (3.10×)	48.6 (1.93×)	78.8 (1.19×)	43.4 (2.16×)
				8	101.3	76.9 (1.32×)	32.4 (3.13×)	32.3 (3.14×)	30.7 (3.31×)	65.8 (1.54×)	142.7 (0.71×)	43.7 (2.32×)
				16	105.9	147.3 (0.72×)	32.4 (3.26×)	33.3 (3.18×)	30.7 (3.45×)	190.0 (0.56×)	274.4 (0.39×)	45.0 (2.35×)
				32	98.3	281.0 (0.35×)	48.9 (2.01×)	46.6 (2.11×)	46.8 (2.10×)	185.3 (0.53×)	531.3 (0.18×)	46.0 (2.13×)
				64	100.1	552.3 (0.18×)	77.7 (1.29×)	75.4 (1.33×)	76.0 (1.32×)	184.6 (0.54×)	1048.3 (0.10×)	83.6 (1.20×)
				128	124.6	1091.6 (0.11×)	115.4 (1.08×)	111.9 (1.11×)	113.0 (1.10×)	213.7 (0.58×)	2083.7 (0.06×)	106.5 (1.17×)
Qwen-3-32B	attn.qkv	5120	10240	1	85.3	23.3 (3.66×)	28.2 (3.03×)	27.6 (3.09×)	27.7 (3.08×)	30.5 (2.80×)	26.6 (3.21×)	28.2 (3.03×)
				2	85.3	26.4 (3.23×)	28.1 (3.03×)	27.0 (3.16×)	27.6 (3.09×)	28.0 (3.05×)	40.8 (2.09×)	28.8 (2.97×)
				4	85.3	37.7 (2.26×)	29.0 (2.94×)	26.6 (3.21×)	27.5 (3.10×)	48.6 (1.76×)	69.9 (1.22×)	28.8 (2.96×)
				8	85.2	66.4 (1.28×)	28.2 (3.03×)	27.4 (3.11×)	27.7 (3.08×)	71.8 (1.19×)	127.8 (0.67×)	28.2 (3.02×)
				16	86.5	130.1 (0.66×)	28.3 (3.06×)	27.0 (3.20×)	27.6 (3.13×)	157.4 (0.55×)	241.2 (0.36×)	29.6 (2.92×)
				32	89.2	245.9 (0.36×)	35.0 (2.54×)	34.4 (2.59×)	33.5 (2.66×)	157.3 (0.57×)	473.7 (0.19×)	39.8 (2.24×)
				64	89.3	481.2 (0.19×)	60.7 (1.47×)	56.8 (1.57×)	56.9 (1.57×)	159.0 (0.56×)	931.3 (0.10×)	57.9 (1.54×)
				128	95.9	908.9 (0.11×)	98.0 (0.98×)	93.5 (1.03×)	94.2 (1.02×)	166.3 (0.58×)	1851.3 (0.05×)	86.7 (1.11×)
	attn.o	8192	5120	1	67.0	19.3 (3.47×)	25.1 (2.67×)	24.0 (2.79×)	24.4 (2.75×)	21.4 (3.13×)	22.9 (2.92×)	28.2 (2.37×)
				2	66.1	22.3 (2.96×)	25.1 (2.64×)	24.2 (2.74×)	24.1 (2.74×)	26.9 (2.46×)	34.3 (1.93×)	28.1 (2.35×)
				4	66.2	30.1 (2.20×)	25.5 (2.60×)	24.8 (2.67×)	24.6 (2.69×)	32.9 (2.01×)	59.1 (1.12×)	28.2 (2.35×)
				8	66.8	54.9 (1.22×)	25.2 (2.65×)	25.0 (2.67×)	24.6 (2.72×)	45.9 (1.46×)	107.2 (0.62×)	28.4 (2.35×)
				16	70.0	103.3 (0.68×)	25.5 (2.74×)	25.3 (2.76×)	24.9 (2.81×)	118.8 (0.59×)	201.6 (0.35×)	29.3 (2.39×)
				32	69.3	197.5 (0.35×)	37.1 (1.87×)	36.4 (1.91×)	35.7 (1.94×)	117.6 (0.59×)	390.0 (0.18×)	30.8 (2.25×)
				64	70.2	391.1 (0.18×)	58.3 (1.21×)	55.9 (1.26×)	57.0 (1.23×)	120.5 (0.58×)	771.0 (0.09×)	50.3 (1.40×)
				128	77.3	763.1 (0.10×)	83.3 (0.93×)	81.2 (0.95×)	81.6 (0.95×)	125.9 (0.61×)	1526.8 (0.05×)	66.0 (1.17×)
	mlp.gateup	5120	51200	1	394.2	103.8 (3.80×)	113.7 (3.47×)	109.6 (3.60×)	110.2 (3.58×)	108.9 (3.62×)	110.8 (3.56×)	119.5 (3.30×)
				2	397.0	110.6 (3.59×)	115.7 (3.43×)	113.0 (3.51×)	109.2 (3.64×)	122.7 (3.23×)	179.3 (2.21×)	119.5 (3.32×)
				4	398.7	167.0 (2.39×)	114.9 (3.47×)	110.6 (3.61×)	109.4 (3.64×)	173.6 (2.30×)	318.7 (1.25×)	119.7 (3.33×)
				8	398.8	324.2 (1.23×)	114.7 (3.48×)	110.2 (3.62×)	111.2 (3.59×)	313.8 (1.27×)	585.6 (0.68×)	119.9 (3.33×)
				16	399.3	584.8 (0.68×)	112.3 (3.56×)	111.0 (3.60×)	108.2 (3.69×)	886.2 (0.45×)	1144.8 (0.35×)	122.7 (3.25×)
				32	404.2	1142.0 (0.35×)	131.8 (3.07×)	121.9 (3.32×)	120.4 (3.36×)	892.4 (0.45×)	2234.6 (0.18×)	204.4 (1.98×)
				64	422.5	2212.9 (0.19×)	231.9 (1.82×)	214.5 (1.97×)	223.6 (1.89×)	908.1 (0.47×)	4424.1 (0.10×)	415.9 (1.02×)
				128	464.7	4348.6 (0.11×)	434.7 (1.07×)	417.8 (1.11×)	420.1 (1.11×)	950.8 (0.49×)	8812.7 (0.05×)	451.8 (1.03×)
	mlp.down	25600	5120	1	202.3	57.0 (3.55×)	60.7 (3.33×)	59.3 (3.41×)	64.4 (3.14×)	54.9 (3.69×)	57.4 (3.53×)	73.2 (2.76×)
				2	200.3	56.9 (3.52×)	58.8 (3.41×)	59.0 (3.39×)	65.6 (3.06×)	68.9 (2.91×)	93.5 (2.14×)	74.2 (2.70×)
				4	201.3	86.8 (2.32×)	60.1 (3.35×)	58.6 (3.43×)	64.5 (3.12×)	79.6 (2.53×)	165.3 (1.22×)	73.2 (2.75×)
				8	202.0	162.2 (1.25×)	59.4 (3.40×)	60.1 (3.36×)	64.6 (3.13×)	107.8 (1.87×)	299.6 (0.67×)	74.1 (2.73×)
				16	209.5	308.4 (0.68×)	60.8 (3.45×)	60.7 (3.45×)	64.0 (3.27×)	447.6 (0.47×)	581.9 (0.36×)	74.7 (2.80×)
				32	206.9	609.4 (0.34×)	80.1 (2.58×)	75.4 (2.74×)	74.1 (2.79×)	444.1 (0.47×)	1136.9 (0.18×)	83.4 (2.48×)
				64	211.4	1193.0 (0.18×)	133.3 (1.59×)	129.3 (1.64×)	129.1 (1.64×)	447.3 (0.47×)	2258.0 (0.09×)	154.7 (1.37×)
				128	234.3	2369.5 (0.10×)	231.0 (1.01×)	221.2 (1.06×)	224.4 (1.04×)	463.7 (0.51×)	4477.7 (0.05×)	196.7 (1.19×)

Table 17: Kernel latency microbenchmark (μs, speedup over FP16) (DGX SPARK)

Model	Layer	K	N	M	FP16	RaZeR-CUDA	RaZeR-TC	Marlin	Marlin-FP4	Any-Precision	SqueezeLLM	AWQ
Llama-3.1-8B	attn.qkv	4096	6144	1	374.7	85.2 (4.40×)	68.8 (5.44×)	68.6 (5.46×)	67.3 (5.57×)	76.5 (4.89×)	70.8 (5.29×)	74.5 (5.03×)
				2	263.8	80.1 (3.29×)	83.0 (3.18×)	86.2 (3.06×)	76.8 (3.44×)	87.6 (3.01×)	89.7 (2.94×)	74.6 (3.54×)
				4	258.0	84.0 (3.07×)	66.5 (3.88×)	66.0 (3.91×)	65.5 (3.94×)	132.6 (1.95×)	144.7 (1.78×)	77.3 (3.34×)
				8	257.5	153.8 (1.67×)	78.3 (3.29×)	75.1 (3.43×)	68.1 (3.78×)	200.6 (1.28×)	242.1 (1.06×)	72.9 (3.53×)
				16	263.6	275.4 (0.96×)	67.8 (3.89×)	76.7 (3.44×)	69.1 (3.82×)	555.1 (0.47×)	412.6 (0.64×)	72.2 (3.65×)
				32	319.4	449.5 (0.71×)	72.2 (4.43×)	70.9 (4.50×)	68.9 (4.64×)	616.3 (0.52×)	761.8 (0.42×)	107.1 (2.98×)
				64	243.3	796.4 (0.31×)	79.1 (3.08×)	80.0 (3.04×)	76.0 (3.20×)	542.2 (0.45×)	1511.6 (0.16×)	131.4 (1.85×)
				128	358.8	1491.2 (0.24×)	124.4 (2.88×)	111.5 (3.22×)	112.0 (3.20×)	615.5 (0.58×)	2896.2 (0.12×)	224.3 (1.60×)
	attn.o	4096	4096	1	238.4	50.5 (4.72×)	56.6 (4.21×)	47.6 (5.01×)	51.5 (4.63×)	48.0 (4.97×)	52.3 (4.56×)	55.2 (4.32×)
				2	194.8	55.0 (3.54×)	52.9 (3.69×)	50.6 (3.85×)	57.6 (3.38×)	70.7 (2.75×)	72.8 (2.67×)	55.7 (3.49×)
				4	163.3	53.5 (3.05×)	46.7 (3.50×)	51.5 (3.17×)	51.4 (3.18×)	93.3 (1.75×)	111.3 (1.47×)	54.3 (3.01×)
				8	162.0	113.1 (1.43×)	49.2 (3.30×)	51.4 (3.15×)	49.2 (3.29×)	102.3 (1.58×)	169.8 (0.95×)	62.1 (2.61×)
				16	182.6	212.5 (0.86×)	52.0 (3.51×)	49.3 (3.70×)	53.0 (3.45×)	354.0 (0.52×)	304.1 (0.60×)	65.1 (2.80×)
				32	208.1	346.7 (0.60×)	63.1 (3.30×)	53.3 (3.91×)	56.3 (3.69×)	363.8 (0.57×)	537.4 (0.39×)	71.8 (2.90×)
				64	218.4	587.2 (0.37×)	62.2 (3.51×)	57.3 (3.81×)	65.5 (3.33×)	413.0 (0.53×)	1027.0 (0.21×)	105.6 (2.07×)
				128	213.6	1030.7 (0.21×)	116.4 (1.83×)	83.8 (2.55×)	94.6 (2.26×)	403.9 (0.53×)	2038.6 (0.10×)	133.5 (1.60×)
	mlp.gateup	4096	28672	1	1083.6	302.4 (3.58×)	306.4 (3.54×)	302.4 (3.58×)	297.9 (3.64×)	304.4 (3.56×)	339.3 (3.19×)	363.1 (2.98×)
				2	1105.9	361.9 (3.06×)	302.0 (3.66×)	321.1 (3.44×)	299.6 (3.69×)	363.8 (3.04×)	372.6 (2.97×)	355.1 (3.11×)
				4	1141.3	439.5 (2.60×)	297.5 (3.84×)	323.6 (3.53×)	303.9 (3.76×)	429.3 (2.66×)	533.8 (2.14×)	318.4 (3.58×)
				8	1095.7	649.8 (1.69×)	309.0 (3.55×)	303.6 (3.61×)	315.4 (3.47×)	749.4 (1.46×)	910.7 (1.20×)	368.4 (2.97×)
				16	1101.2	1052.3 (1.05×)	326.7 (3.37×)	307.1 (3.59×)	298.7 (3.69×)	2512.7 (0.44×)	1720.1 (0.64×)	338.4 (3.25×)
				32	1254.9	1863.4 (0.67×)	322.9 (3.89×)	334.3 (3.75×)	282.0 (4.45×)	2732.3 (0.46×)	3340.8 (0.38×)	608.9 (2.06×)
				64	1288.1	3488.7 (0.37×)	355.1 (3.63×)	349.3 (3.69×)	324.4 (3.97×)	2794.3 (0.46×)	6609.9 (0.19×)	1150.0 (1.12×)
				128	1315.8	6658.6 (0.20×)	491.5 (2.68×)	439.5 (2.99×)	460.0 (2.86×)	2776.2 (0.47×)	13083.4 (0.10×)	1210.8 (1.09×)
	mlp.down	14336	4096	1	590.2	164.2 (3.59×)	147.7 (4.00×)	183.0 (3.22×)	149.3 (3.95×)	174.4 (3.38×)	167.4 (3.53×)	213.7 (2.76×)
				2	611.8	174.2 (3.51×)	155.6 (3.93×)	175.7 (3.48×)	157.9 (3.87×)	226.4 (2.70×)	210.5 (2.91×)	182.1 (3.36×)
				4	580.9	219.7 (2.64×)	162.5 (3.58×)	183.3 (3.17×)	149.3 (3.89×)	250.4 (2.32×)	303.9 (1.91×)	221.3 (2.63×)
				8	585.5	331.9 (1.76×)	170.0 (3.44×)	141.3 (4.14×)	149.4 (3.92×)	301.0 (1.95×)	518.0 (1.13×)	190.4 (3.08×)
				16	581.0	614.1 (0.95×)	163.6 (3.55×)	155.2 (3.74×)	178.6 (3.25×)	1315.8 (0.44×)	920.8 (0.63×)	214.9 (2.70×)
				32	677.5	1002.3 (0.68×)	165.2 (4.10×)	163.6 (4.14×)	160.1 (4.23×)	1390.2 (0.49×)	1698.4 (0.40×)	232.4 (2.92×)
				64	710.6	1808.1 (0.39×)	209.3 (3.40×)	200.8 (3.54×)	178.7 (3.98×)	1468.1 (0.48×)	3325.7 (0.21×)	376.9 (1.89×)
				128	728.6	3503.1 (0.21×)	291.1 (2.50×)	246.9 (2.95×)	255.3 (2.85×)	1468.9 (0.50×)	6593.6 (0.11×)	508.4 (1.43×)
Qwen-3-32B	attn.qkv	5120	10240	1	556.0	182.9 (3.04×)	138.3 (4.02×)	159.6 (3.48×)	150.0 (3.71×)	159.7 (3.48×)	153.5 (3.62×)	171.7 (3.24×)
				2	554.8	166.6 (3.33×)	161.9 (3.43×)	133.3 (4.16×)	133.3 (4.16×)	176.1 (3.15×)	185.9 (2.98×)	168.3 (3.30×)
				4	534.1	176.6 (3.02×)	148.1 (3.61×)	137.4 (3.89×)	136.3 (3.92×)	255.7 (2.09×)	267.5 (2.00×)	182.0 (2.93×)
				8	553.0	260.9 (2.12×)	139.6 (3.96×)	158.1 (3.50×)	129.2 (4.28×)	331.5 (1.67×)	446.9 (1.24×)	179.5 (3.08×)
				16	553.5	573.4 (0.97×)	180.3 (3.07×)	144.5 (3.83×)	131.9 (4.20×)	1556.4 (0.36×)	797.5 (0.69×)	168.5 (3.28×)
				32	607.3	936.3 (0.65×)	163.9 (3.71×)	135.1 (4.50×)	141.0 (4.31×)	1294.1 (0.47×)	1524.5 (0.40×)	221.1 (2.75×)
				64	649.2	1579.7 (0.41×)	140.9 (4.61×)	144.4 (4.50×)	163.7 (3.97×)	1328.8 (0.49×)	3002.6 (0.22×)	397.4 (1.63×)
				128	677.7	2998.5 (0.23×)	256.4 (2.64×)	229.6 (2.95×)	219.9 (3.08×)	1321.0 (0.51×)	5919.0 (0.11×)	451.2 (1.50×)
	attn.o	8192	5120	1	431.5	106.4 (4.05×)	116.3 (3.71×)	109.2 (3.95×)	111.7 (3.86×)	127.9 (3.37×)	113.2 (3.81×)	135.8 (3.18×)
				2	396.8	120.2 (3.30×)	122.7 (3.24×)	107.1 (3.70×)	108.4 (3.66×)	155.7 (2.55×)	150.3 (2.64×)	128.8 (3.08×)
				4	438.9	137.9 (3.18×)	112.8 (3.89×)	105.2 (4.17×)	107.9 (4.07×)	216.4 (2.03×)	235.4 (1.86×)	110.5 (3.97×)
				8	397.9	254.9 (1.56×)	121.4 (3.28×)	109.4 (3.64×)	105.3 (3.78×)	267.6 (1.49×)	357.9 (1.11×)	116.1 (3.43×)
				16	442.0	482.6 (0.92×)	114.7 (3.85×)	110.7 (3.99×)	109.4 (4.04×)	930.6 (0.47×)	654.5 (0.68×)	127.3 (3.47×)
				32	490.5	735.4 (0.67×)	124.6 (3.94×)	112.0 (4.38×)	106.4 (4.61×)	997.4 (0.49×)	1231.3 (0.40×)	169.7 (2.89×)
				64	483.9	1285.8 (0.38×)	124.3 (3.89×)	119.5 (4.05×)	117.2 (4.13×)	1089.1 (0.44×)	2406.8 (0.20×)	261.8 (1.85×)
				128	554.7	2453.3 (0.23×)	224.6 (2.47×)	192.3 (2.89×)	195.0 (2.85×)	1067.5 (0.52×)	4788.1 (0.12×)	360.2 (1.54×)
	mlp.gateup	5120	51200	1	2461.8	656.3 (3.75×)	669.8 (3.68×)	667.2 (3.69×)	684.6 (3.60×)	693.8 (3.55×)	675.8 (3.64×)	712.0 (3.46×)
				2	2386.8	700.1 (3.41×)	689.5 (3.46×)	640.1 (3.73×)	648.2 (3.68×)	739.1 (3.23×)	830.6 (2.87×)	737.1 (3.24×)
				4	2463.6	860.4 (2.86×)	673.6 (3.66×)	643.7 (3.83×)	651.3 (3.78×)	810.4 (3.04×)	1122.9 (2.19×)	723.2 (3.41×)
				8	2416.3	1137.2 (2.12×)	689.6 (3.50×)	644.8 (3.75×)	644.2 (3.75×)	1215.3 (1.99×)	1992.8 (1.21×)	722.9 (3.34×)
				16	2419.6	2003.3 (1.21×)	660.4 (3.66×)	678.9 (3.56×)	684.2 (3.54×)	5558.8 (0.44×)	3789.3 (0.64×)	739.5 (3.27×)
				32	2770.9	3871.8 (0.72×)	729.3 (3.80×)	689.6 (4.02×)	694.8 (3.99×)	5960.8 (0.46×)	7362.3 (0.38×)	1452.8 (1.91×)
				64	2859.2	7412.2 (0.39×)	712.1 (4.02×)	721.6 (3.96×)	733.7 (3.90×)	6016.6 (0.48×)	14664.7 (0.19×)	2535.5 (1.13×)
				128	2959.2	14573.9 (0.20×)	936.7 (3.16×)	921.4 (3.21×)	919.7 (3.22×)	5983.2 (0.49×)	29205.2 (0.10×)	2634.8 (1.12×)
	mlp.down	25600	5120	1	1251.2	361.1 (3.47×)	368.3 (3.40×)	320.7 (3.90×)	327.4 (3.82×)	385.2 (3.25×)	368.2 (3.40×)	405.2 (3.09×)
				2	1209.1	340.9 (3.55×)	385.7 (3.13×)	325.3 (3.72×)	319.1 (3.79×)	424.6 (2.85×)	395.5 (3.06×)	402.2 (3.01×)
				4	1216.3	398.6 (3.05×)	381.2 (3.19×)	326.0 (3.73×)	331.7 (3.67×)	447.5 (2.72×)	577.3 (2.11×)	391.1 (3.11×)
				8	1250.4	677.1 (1.85×)	375.1 (3.33×)	339.0 (3.69×)	320.4 (3.90×)	521.8 (2.40×)	1000.4 (1.25×)	357.5 (3.50×)
				16	1298.2	1113.5 (1.17×)	377.9 (3.44×)	346.1 (3.75×)	356.5 (3.64×)	2949.3 (0.44×)	1941.7 (0.67×)	401.4 (3.23×)
				32	1500.5	2165.2 (0.69×)	374.5 (4.01×)	332.4 (4.51×)	331.7 (4.52×)	3131.2 (0.48×)	3744.7 (0.40×)	633.6 (2.37×)
				64	1548.6	3905.3 (0.40×)	390.6 (3.96×)	369.9 (4.19×)	356.2 (4.35×)	3248.4 (0.48×)	7351.0 (0.21×)	733.7 (2.11×)
				128	1643.0	7506.5 (0.22×)	504.4 (3.26×)	473.4 (3.47×)	508.9 (3.23×)	3341.2 (0.49×)	14566.4 (0.11×)	1301.2 (1.26×)

E Additional Experiments on Auto-tuned Kernel Performance

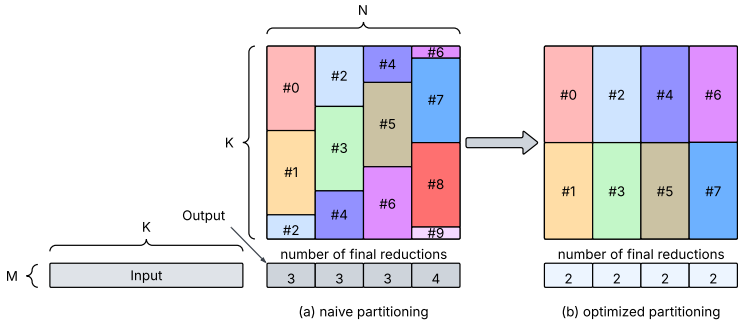


Figure 6: Optimized partitioning for small weight tensors. The optimized strategy reduces the number of global reduction stages and associated overhead.

In the RaZeR weight-only tensor-core kernel, the weight matrix is partitioned into multiple stripes of approximately equal length, potentially spanning multiple N-dimensions. Each stripe's length must be a multiple of 256 to satisfy low-level kernel design constraints. Each thread block fully occupies one SM, processes one stripe, and accumulates partial results to global memory.

On modern GPUs with a large number of SMs, this strategy can excessively partition small matrices. We observe that in the memory-bound scenario, reducing the number of active SMs does not underutilize memory bandwidth. Motivated by this observation, we introduce an auto-tuning strategy that reduces the number of partitions and global reduction stages to minimize GEMM and GEMV latency.

Figure 6 illustrates the approach. For a GPU with 10 SMs, the naive strategy utilizes all SMs, resulting in 3–4 reduction stages per output tile. In contrast, the optimized strategy launches only 8 SMs, reducing the number of global reductions and improving GEMM/GEMV latency. Table 18 reports profiling results on an NVIDIA RTX 5090 (170 SMs), where the auto-tuned kernel achieves up to 9.87% throughput improvement.

Table 18: End-to-end inference decode speed measurements on RTX 5090 for several small models.

Model	Batch Size	Default (tok/s)	Auto-tuned (tok/s)	Improvement
Llama-3.2-1B	1	611.04	653.83	7.00%
	2	610.40	652.23	6.85%
	4	604.65	644.40	6.57%
	8	603.85	628.73	4.12%
	16	578.74	620.68	7.25%
	32	430.81	473.31	9.87%
	64	333.40	365.10	9.51%
Llama-3.2-3B	1	329.40	338.95	2.90%
	2	323.05	337.90	4.60%
	4	319.45	340.25	6.51%
	8	296.94	317.18	6.82%
	16	268.69	274.85	2.29%
	32	190.57	199.68	4.78%
	64	128.70	129.39	0.54%
Llama-3.1-8B	1	222.133	233.18	4.97%
	2	216.79	222.06	2.43%
	4	215.46	227.00	5.36%
	8	209.43	225.67	7.75%
	16	207.24	219.93	6.12%
	32	170.03	170.70	0.39%
	64	117.15	118.24	0.93%

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims presented in the abstract and introduction accurately reflect the contributions and scope of the paper. The contributions are well-supported by experimental results, demonstrating both validity and credibility.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a dedicated "Limitations" paragraph at the end, which acknowledges two limitations: the need of future work to more accurately quantify the hardware overhead, and the weight block scale may still require E4M3 for some LLMs in practice.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The claim of redundant sign bit in the E4M3 block scale is supported by complete, step-by-step math derivations.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper gives full details for reproduction, including baseline models, datasets, and special values used. It also provides clear explanation about how the RaZeR kernel is implemented.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Our evaluation uses public models and datasets. We provide the public code repository, linked in the abstract and the appendix.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5.1 provides complete details about the experimental setup, including evaluated models, datasets, and baseline methods.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We conduct comprehensive accuracy evaluation on different LLMs and datasets, as listed in Table 2, Table 3, and Table 4. Our kernel evaluation covers a wide range of GPUs, batch size and matrix size, as shown in Fig. 4 , Fig. 5, Table 16, Table 15, and Table 17.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In Section 5.3, we describe what GPUs (NVIDIA RTX Pro 6000 Server Edition, DGX Spark, and RTX 5090) are used for kernel performance evaluation.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: All research conducted in our paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

Justification: The proposed RaZeR format pushes the limit of NVFP4 to a new state-of-the-art. We conduct comprehensive evaluations to quantify the accuracy benefits and hardware performance of RaZeR. Thus, we believe this work can motivate future research on designing more efficient quantization formats.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release new models or datasets. It builds on existing models (e.g., Llama and Qwen) and public datasets, which do not require additional safeguards beyond those already established by their original authors.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All models and datasets used in our evaluation are open-sourced. We properly cite the original paper / link of all models, datasets, and methods compared.

Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The proposed quantization format, RaZeR, is well documented. The paper provides detailed theoretical foundations, experimental settings, and evaluations.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

Answer: [N/A]

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

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: Our method does not involve LLMs as any important, original, or non-standard components.

Guidelines:

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