QUANTAR: QUANTIZING ALL-REDUCE FOR EFFICIENT TENSOR-PARALLEL SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization has reached nearly every tensor a language model stores or computes, yet tensor-parallel all-reduce still communicates in BF16. We find that this communication accounts for as much as 37% of prefill device time across models ranging from an 8B dense transformer to a 671B mixture-of-experts model, and its share grows as lower-precision computation makes the rest of inference faster. We introduce **Quantar** to quantize the reduction itself. Existing dynamic quantization, if applied to all-reduce, would require an extra communication round for scale agreement; without it, the reduced values cannot be mapped back to the correct sum. Quantar instead freezes communication scales calibrated once and shares them across ranks and requests. We show that transformer normalization places an input-independent bound on all-reduce activations, making one frozen scale practical across ranks. On eight B300 GPUs, Quantar speeds up the complete communication boundary by up to $1.81\times$ and prefill by up to $1.28\times$, and on eight RTX 6000 Pros, performance measurements reach $1.92\times$ and $1.61\times$. At matched settings, Quantar outperforms per-hop FP8 by up to $1.24\times$ at the communication boundary and $1.10\times$ in prefill, while avoiding its $1.54\times$ error growth from two to eight ranks. We also show that quality remains at BF16 parity: full 12,032-question MMLU-Pro scores change by at most 0.42 percentage points across five models, and all nine benchmarks and 33 subtasks evaluated across three models remain within 2σ. Quantar also matches BF16 on LongBench-E, GSM8K-CoT, and RULER from 8k through 128k, including 85.4 versus 85.0 at 128k. Quantar's advantage grows as models become larger, more sharded, and more heavily quantized, extending low-bit gains across the inference path.

1 INTRODUCTION

Prefill is a critical phase of large language model inference, with much of its cost arising from matrix multiplications. With optimized kernels, these matrix multiplications are largely compute-bound, leaving two main levers to make them faster: faster hardware with higher FLOPs or quantization. Quantization lets us represent weights and activations in lower precision, allowing hardware to process more values in one pass. Examples include post-training methods such as GPTQ, QuaRot, and SmoothQuant (Frantar et al., 2023; Ashkboos et al., 2024; Xiao et al., 2023), and low-precision formats such as FP8 (Micikevicius et al., 2022).

As large language models have grown, Megatron-style tensor parallelism has become a standard way to fit and serve them across GPUs with lower latency and higher throughput (Shoeybi et al., 2019). It shards the computation of each sublayer (attention or MLP/MoE) across GPU ranks, and after each sublayer's output projection, All-Reduce sums the partial outputs before they are added to the residual stream. Quantization shortens the computation surrounding the All-Reduce, so communication occupies a larger fraction of prefill and becomes the next bottleneck. In our profiling, moving DeepSeek-V3.1 (DeepSeek-AI, 2025) from FP8 to NVIDIA's NVFP4 checkpoint (NVIDIA, 2025a) raises the All-Reduce share of prefill device time from 29% to 37%.

So, the next lever is to quantize GPU communication. A natural approach is to treat the communicated partial tensors as activations and apply the dynamic quantization used by QuickReduce and FlashInfer; a similar vLLM path remains under review (Li et al., 2025a; FlashInfer contributors, 2026; matkle, 2026) (Section A). Dynamic quantization measures the current activation range on

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

each rank, so the ranks express their values in different units. Prior work therefore either reconstructs every contribution before summing or rounds the running low-precision sum at each hop (Section 2.3). The core problem is to find one scale valid across all ranks and requests, without measuring and synchronizing their ranges during every All-Reduce.

Therefore, we introduce Quantar and show that the communicated tensor's activation range does not need to be computed during inference: pre-norm transformers normalize the input to every sublayer, which *bounds* what each GPU sends regardless of the input. This lets Quantar freeze the quantization offline, reuse it for every request and every GPU, and quantize the reduction itself without runtime scale agreement or repeated low-precision accumulation. The shared scale also makes in-switch reduction through NVLink SHARP directly usable (NVIDIA, 2026c;a).

Quantar matches BF16 quality across six models and contexts up to 128k on RULER, GSM8K-CoT, MMLU-Pro, and other tasks. On eight B300 GPUs, it speeds up the All-Reduce by up to $1.81\times$ and prefill by up to $1.28\times$. These gains grow as lower-precision computation makes communication a larger share of prefill.

2 Background and Related Work

2.1 Post-Training Activation Quantization

Post-training quantization methods differ in which tensors they quantize. Weight-only methods such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) reduce model storage and the memory traffic during the forward pass. Methods such as LLM.int8() (Dettmers et al., 2022), SmoothQuant (Xiao et al., 2023), and QuaRot (Ashkboos et al., 2024) also quantize activations, allowing matrix multiplications to use low-precision arithmetic. Our work likewise concerns activations, but in their role as the partial outputs communicated between GPUs.

Following quantization literature, we write activation quantization as

$$q = \mathcal{Q}(x/s), \qquad \widehat{x} = s\,q, \qquad s = \frac{\max|x|}{C},$$

where $s > 0$ is the scale, C is the largest magnitude represented by the format, and $\mathcal{Q}$ rounds to its nearest representable value. The scale s maps x onto the format's representable grid, trading the largest magnitude it can capture against the spacing between representable values. A scale may be shared by the whole tensor or by a smaller group of values. For an activation $\mathbf{X} \in \mathbb{R}^{T\times d}$, per-tensor quantization uses one scale for all of $\mathbf{X}$, while per-token and per-channel quantization use one scale for each row and column. Per-block formats divide a row into consecutive groups: MX formats share one scale across 32 values (Open Compute Project, 2023), while NVFP4 uses an E4M3 scale for every 16 values and one FP32 scale for the complete tensor (NVIDIA, 2025b). Smaller groups follow local ranges more closely but store more scales.

An activation scale is *static* when estimated offline from calibration data and *dynamic* when computed from the current activation (Xiao et al., 2023). LLM.int8() and QuaRot use dynamic per-token scales (Dettmers et al., 2022; Ashkboos et al., 2024); SmoothQuant supports static per-tensor and dynamic per-token variants (Xiao et al., 2023); and DeepSeek-V3 uses dynamic scales for 128-channel blocks of each token (DeepSeek-AI, 2024). These methods select scales for a GPU's local matrix multiplication. For an All-Reduce, the relevant question is whether the partial activations have a scale of their own, or a global shared scale.

2.2 Collective Communication in Tensor Parallelism

Suppose tensor-parallel rank r produces a partial output $\mathbf{x}^{(r)}$. The All-Reduce returns

$$\mathbf{x} = \textstyle\sum_{r=1}^{W} \mathbf{x}^{(r)}$$

to every rank before the model continues (Shoeybi et al., 2019). A common implementation separates this operation into two phases. During *reduce-scatter*, the ranks exchange and sum disjoint slices of their partial outputs, leaving each rank with one slice of $\mathbf{x}$. During *all-gather*, they exchange these completed slices so that every rank recovers the full result (Thakur et al., 2005).

Collective algorithms differ in the path these slices take and in where their contributions are added. A ring, for example, sends slices from rank to rank and one-by-one adds up at each step (Patarasuk & Yuan, 2009). Other implementations send contributions directly to the GPU responsible for each output slice before distributing the completed result.

The distinction relevant here is the arithmetic along that path: repeatedly accumulating in low precision rounds the running sum after each addition, whereas reconstructing the partials before accumulation avoids repeated low-precision rounding but adds conversion and data-movement overhead.

Recent HGX systems can instead reduce inside the NVSwitch fabric through NVLink SHARP (NVLS) (NVIDIA, 2026a;c). A GPU issues a multicast-memory reduction load for one output partition; NVSwitch combines the corresponding rank contributions, and the requesting GPU multicasts the completed partition to the other ranks. The switch adds the supplied values but cannot adjust one sender's values to another sender's scale. A direct low-precision sum therefore requires every rank to use the same scale.

2.3 Quantized Communication

Inference. Several prior inference systems reconstruct each partial before summation. Flash Communication and its successor (Li et al., 2024; 2025b), QuickReduce (Li et al., 2025a), the method of Hansen-Palmus et al. (2024), and FlashInfer's quantized All-Reduce (FlashInfer contributors, 2026) measure a range per rank, communicate the compressed partial, and recover its original unit before accumulating. A related vLLM implementation was under review in GitHub PR #39783 as of September 25, 2026 (matkle, 2026). Dong et al. (2024) calibrate the rank-local scales offline, but still all-gather the partials and sum them in BF16. They evaluate compression and downstream accuracy, but do not implement the communication system or report latency or throughput, leaving system-level efficiency to future work. NCCL exposes FP8 All-Reduce (NVIDIA, 2026a). In our NCCL 2.28.9 installation, the default path rounds intermediate sums back to FP8 (Section A). SCIN (Jiang et al., 2026) adds scale-aware arithmetic to the switch, but requires new hardware. These designs leave a choice between reconstructing rank-local units before the sum and repeatedly rounding the sum itself.

Training. Communication compression has a longer history in training. 1-bit SGD and QSGD quantize gradient updates (Seide et al., 2014; Alistarh et al., 2017), whereas PowerSGD communicates low-rank gradient factors (Vogels et al., 2019). Error feedback carries compression residuals into later steps (Karimireddy et al., 2019). ZeRO++ (Wang et al., 2024a) quantizes gradients and weights in blocks and recovers full precision before the reduction. Closer to our setting, TACO (Liu et al., 2026) communicates FP8 tensor-parallel intermediates during training using adaptive reshaping, a Scale–Hadamard transform, and dual-scale quantization.

Rather than making rank-local scaling cheaper, Quantar removes the need for it. It fixes one per-channel scale across requests and ranks, so the scale factors out and the partials can be reduced directly in low precision (Section 3). SCIN changes the switch to accommodate rank-local scales; Quantar makes the values compatible before they enter the collective. Existing NVLS hardware can therefore reduce them in-switch without any scale-aware extension. The next section shows why the scale can remain fixed.

3 Quantar

Quantizing one partial output is straightforward. The difficulty is that an FP8 value has no meaning without its scale. For one output channel, rank r maps its partial value $x^{(r)}$ to

$$q^{(r)} = \mathcal{Q}\left(x^{(r)}/s^{(r)}\right),$$

and reconstructs that contribution as $s^{(r)}q^{(r)}$. The desired output is therefore $\sum_r s^{(r)}q^{(r)}$, not merely $\sum_r q^{(r)}$.

This difference cannot be repaired after the reduction. Let two ranks use $s^{(1)} = 1$ and $s^{(2)} = 2$. If the ranks send $(q^{(1)}, q^{(2)}) = (2, 0)$, their reconstructed sum is 2; if they send $(0, 2)$, it is 4. All-Reduce returns the same quantized sum, 2, in both cases. That sum no longer records which rank

supplied the nonzero value, so there is no way to know which rank's scale should reconstruct it. Direct reduction therefore requires every rank to use the same scale for a channel.

Quantar uses one shared scale for each output channel. Channel ranges differ, so separate channel scales preserve their resolution; sharing each scale across tokens and ranks keeps the reduction valid. At All-Reduce site ℓ, let $\mathbf{x}_\ell^{(r)}$ be rank r's partial output, $\mathbf{q}_\ell^{(r)}$ its quantized value, and $\mathbf{s}_\ell$ the scale shared by all ranks. The shared scale factors out of the sum:

$$\mathbf{s}_\ell \odot \sum_{r=1}^{W} \mathbf{q}_\ell^{(r)} = \sum_{r=1}^{W} \left(\mathbf{s}_\ell \odot \mathbf{q}_\ell^{(r)} \right). \tag{1}$$

The collective can thus reduce the quantized values directly and apply $\mathbf{s}_\ell$ once to the completed result.

One way to obtain this scale is to compute it during inference. Every rank would first scan its partial output, then exchange local maxima, then begin the actual data reduction. FP8-LM follows this pattern for gradient All-Reduce by selecting a global scale from the per-GPU scales (Peng et al., 2023). This requires measuring local ranges and synchronizing the resulting scale before the covered collective. Quantar removes both operations from the critical path by fixing the shared scale before inference.

Freezing the scale is usually framed as a calibration gamble. ZeroQuant writes that when token ranges vary, "a static quantization scheme ... would lead to significant accuracy drop" (Yao et al., 2022). SmoothQuant states that "the range of activations is dynamic; it varies for different input samples," and attributes the loss of its static O3 configuration to "the discrepancy between the statically collected statistics and the real evaluation samples' activation statistics" (Xiao et al., 2023). AWQ warns that reconstruction may "overfit the calibration set" and distort features on "out-of-distribution domains" (Lin et al., 2024). Williams and Aletras likewise find "substantial variations in downstream task performance" across calibration data (Williams & Aletras, 2024).

Yet static calibration also succeeds throughout this literature. SmoothQuant collects activation statistics once and reuses them on downstream samples (Xiao et al., 2023). On OPT-6.7B under INT3-g128, AWQ finds that mismatching PubMed and Enron calibration and evaluation data raises perplexity by only 0.5–0.6, compared with 2.3–4.9 for GPTQ (Lin et al., 2024). These results show that a static scale can remain accurate beyond its calibration data, but do not explain when this will hold. The missing distinction is between errors caused by an insufficient range and ordinary rounding inside that range.

As described in Section 2, activation quantization commonly sets the scale from the observed absolute maximum:

$$s = \frac{\max|x|}{C}.$$

This maps the largest observed magnitude to the largest representable magnitude C (Xiao et al., 2023). This makes the current tensor fit inside E4M3's representable range, but it also makes dynamic scaling seem necessary: if a later tensor has a larger maximum, values above sC cannot be represented and must be clipped. That reasoning treats all quantization error as one quantity. Quantar separates the two mechanisms hidden inside it.

For a scalar x, the reconstruction error is

$$e(x; s) = s\mathcal{Q}(x/s) - x.$$

The smallest normal E4M3 magnitude L and largest finite magnitude C set two thresholds, sL and sC. If $|x| > sC$, the value exceeds the format's range; this is *overflow*. Clipping keeps it representable by replacing it with $+sC$ or $-sC$, but loses the excess magnitude. If $0 < |x| < sL$, the value falls into the subnormal range and carries fewer significant bits. At the smallest magnitudes, round-to-nearest returns zero, causing *underflow*. Between the two thresholds, round-to-nearest moves the value to the nearest point on the FP8 grid. The error is at most half the adjacent spacing, which for normal E4M3 is at most $2^{-4} = 6.25\%$ of the value (Section C.5); Micikevicius et al. (2022) define the E4M3 format.

These errors behave differently. Clipping removes excess magnitude, and underflow pulls small values toward zero. Both systematically reduce magnitude, so repeated errors can add instead of

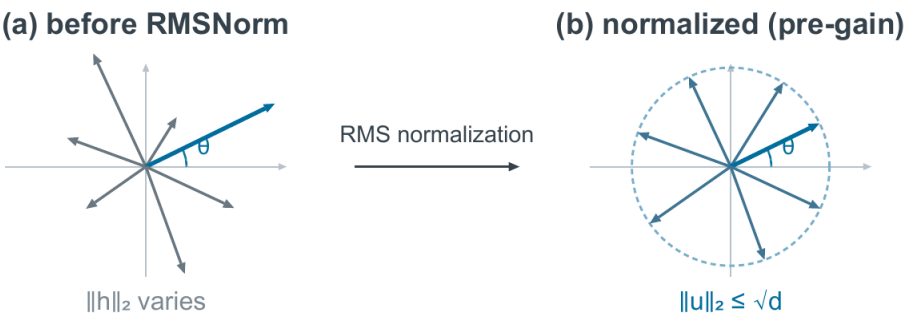


Figure 1: RMSNorm preserves direction while constraining every pre-gain vector to the same norm ceiling. The fixed projection rows then carry this bound to each communicated output channel.

canceling. In-range rounding is instead locally bounded by the adjacent FP8 spacing, and its sign varies with a value's position on the grid. The scale must therefore keep values in range; the rounding left inside that range is a separate accumulation problem.

The first requirement is therefore a request-independent range. A pre-norm transformer already provides the bound needed for it. RMSNorm constrains the input to every attention and MLP or MoE sublayer (Llama Team, AI at Meta, 2024; DeepSeek-AI, 2024; Yang et al., 2025). Zhang & Sennrich (2019) introduced RMSNorm; using the epsilon-stabilized convention of the evaluated checkpoints, for residual stream vector $\mathbf{h} \in \mathbb{R}^d$, learned gain $\boldsymbol{\gamma}$, and $\epsilon > 0$, we write

$$\mathbf{y} = \mathrm{RMSNorm}(\mathbf{h}) = \boldsymbol{\gamma} \odot \frac{\mathbf{h}}{\sqrt{\|\mathbf{h}\|^2/d + \epsilon}}.$$

Before the learned gain, the normalized vector obeys

$$\left\| \frac{\mathbf{h}}{\sqrt{\|\mathbf{h}\|^2/d + \epsilon}} \right\|^2 = \frac{\|\mathbf{h}\|^2}{\|\mathbf{h}\|^2/d + \epsilon} \le d.$$

Hence $\|\mathbf{y}\| \le \|\boldsymbol{\gamma}\|_\infty \sqrt{d}$, independent of the token, request, or magnitude already present in the residual stream.

Cauchy–Schwarz gives the same bound for every fixed projection row. For row $\mathbf{a}$ and bias β,

$$|\langle \mathbf{a}, \mathbf{y} \rangle + \beta| \le \|\mathbf{a}\|\|\mathbf{y}\| + |\beta|.$$

The value, gate, and up projections are therefore bounded by constants fixed with the checkpoint. Attention takes a weighted average of the bounded value vectors: softmax weights are nonnegative and sum to one, so sequence length does not enlarge the bound. In SwiGLU, both projected inputs are bounded and $|\mathrm{SiLU}(u)| \le |u|$, so their product is bounded. Each MoE expert is the same kind of bounded MLP, and the router combines a finite number of expert outputs with bounded weights. Thus the vector $\mathbf{z}_{\ell,t}^{(r)}$ entering rank r's row-parallel output projection obeys $\|\mathbf{z}_{\ell,t}^{(r)}\| \le Z_\ell^{(r)}$, where $Z_\ell^{(r)}$ depends on the weights but not the request.

Applying Cauchy–Schwarz again bounds the tensor Quantar communicates. Let $\mathbf{w}_{\ell,c}^{(r)}$ be the output-projection row for channel c, and let $b_{\ell,c}^{(r)}$ be its bias, or zero when the bias is applied after reduction. Then

$$\left| x_{\ell,t,c}^{(r)} \right| \le \|\mathbf{w}_{\ell,c}^{(r)}\| Z_\ell^{(r)} + \left| b_{\ell,c}^{(r)} \right|. \tag{2}$$

An input changes which channel aligns with a projection row, while the row norm and $Z_\ell^{(r)}$ fix how large that alignment can become. The upper range that absmax searches for at runtime is therefore bounded by the weights.

The key consequence is that each channel has a ceiling fixed by the checkpoint, not the request. A frozen scale is therefore possible before calibration sees any data. Quantar uses calibration to place

E4M3's representable grid around the values each channel actually produces, rather than spreading it across the full worst-case range. It then maps the calibration maximum below C, leaving room for activation ranges to move within the checkpoint-fixed ceiling. Quantar therefore spends some lower-range precision to avoid clipping. As shown below, the signed rounding errors that remain do not pile up across depth. To choose this range, a BF16 pass over corpus $\mathcal{C}$ records, for every site ℓ and channel c, the largest rank-local partial and completed sum:

$$m_{\ell,c} = \max_{t \in \mathcal{C}} \left\{ \max_{r} \left| x^{(r)}_{\ell,t,c} \right|, \left| \sum_{r=1}^{W} x^{(r)}_{\ell,t,c} \right| \right\}.$$

For headroom η, the installed scale is

$$s_{\ell,c} = \frac{\eta m_{\ell,c}}{C}.$$

We choose $\eta = 32$ as a conservative practical setting and hold it fixed throughout our evaluation. This maps the calibration maximum to $C/32$, leaving room for values beyond those observed during calibration. The sum-safety guard in Section C.2 further limits each rank's contribution so that the completed result remains representable. E4M3 has approximately constant relative spacing across its normal range, so increasing the range does not uniformly coarsen values. Additional precision is lost only when small values enter the subnormal range (Micikevicius et al., 2022). Since every channel has its own scale, a large range in one channel does not coarsen another. Cross-corpus evaluation tests how the table handles channel alignments unseen during calibration.

Once values remain in range, the remaining error comes from rounding. Let $\mathbf{x}_\ell = \sum_{r=1}^{W} \mathbf{x}^{(r)}_\ell$ be the unquantized branch output and $\widehat{\mathbf{x}}_\ell$ the value reconstructed after communication. Their difference, $\mathbf{e}_\ell = \widehat{\mathbf{x}}_\ell - \mathbf{x}_\ell$, is the transport error. The residual update becomes

$$\mathbf{h}_{\ell+1} = \mathbf{h}_\ell + \mathbf{x}_\ell + \mathbf{e}_\ell.$$

This difference includes rounding the rank-local partials and the completed sum. Both occur after the current attention or MLP/MoE computation; only the residual addition remains. The transport error therefore enters the stream additively. Later sublayers respond to the perturbed stream, so the replay below holds their branch outputs fixed and isolates the transport error itself.

After N covered sites, let $\mathbf{E}_N = \sum_{\ell=1}^{N} \mathbf{e}_\ell$. Its squared norm is

$$\|\mathbf{E}_N\|^2 = \sum_{\ell=1}^{N} \|\mathbf{e}_\ell\|^2 + 2 \sum_{i<j} \langle \mathbf{e}_i, \mathbf{e}_j \rangle . \tag{3}$$

The first term is the error contributed by each site. Linear pile-up requires the pairwise inner products to remain positive across depth. If the perturbations remain weakly aligned, their pairwise terms cancel and comparable local errors grow as $\sqrt{N}$, not N. Whether deterministic rounding follows this behavior is an empirical question.

The coherence ratio makes this alignment visible:

$$\kappa_N = \frac{\|\mathbf{E}_N\|^2}{\sum_{\ell=1}^{N} \|\mathbf{e}_\ell\|^2}.$$

It is 1 when the pairwise terms cancel and can reach N if the errors align. Across 121 sites, depth could therefore amplify the error energy by as much as $121\times$. We replayed 256 captured rows from one DeepSeek-V3.1 forward with the branch outputs fixed. The measured factor is $\kappa_{121} = 1.114$, and the mean pairwise cosine is 8.10×10^{-4}: the pairwise terms cancel almost exactly. Relative to the accumulated unquantized branch updates, the error peaks at 2.89% and ends at 1.31%. The served evaluations then let every later sublayer process the perturbed stream and answer the closed-loop question, matching BF16 quality (Sections C.6 and 4).

At runtime, each rank quantizes its partial with the stored $\mathbf{s}_\ell$ and the collective reduces the quantized values directly (Eq. (1)), with no activation scan or scale exchange. Because the scale is known before the producer runs, the cast can fuse into the output projection's epilogue, which dynamic scaling cannot do.

Table 1: MMLU-Pro and WikiText-2 on the five checkpoints where full MMLU-Pro was feasible. Parentheses report changes from BF16: percentage points for MMLU-Pro and relative change for perplexity. MMLU-Pro uses all 12,032 questions.

Model	Weights	Communication	MMLU-Pro (%) ↑	WikiText-2 PPL ↓
Llama-3.1-8B	BF16	BF16	36.26	7.937
		Quantar	35.85 (-0.42)	7.982 (+0.58%)
Mistral-Nemo-12B	BF16	BF16	38.94	7.679
		Quantar	38.84 (-0.10)	7.746 (+0.88%)
Qwen3-32B	BF16	BF16	70.39	10.210
		Quantar	70.00 (-0.39)	10.249 (+0.39%)
Nemotron-3-Ultra	BF16	BF16	81.59	3.746
		Quantar	81.45 (-0.14)	3.748 (+0.06%)
DeepSeek-V3.1	NVFP4	BF16	78.25	4.185
		Quantar	77.98 (-0.27)	4.193 (+0.18%)

Table 2: Zero-shot accuracy (%) on the public battery. Avg. is the mean of the eight tasks; Wiki is WikiText-2 word perplexity. The harness variants are WinoGrande-XL and `lambada_openai`.

Model	Method	MMLU-STEM	HellaSwag	ARC-e	ARC-c	BoolQ	PIQA	WinoGrande	LAMBADA	Avg. ↑	Wiki ↓
DeepSeek-V3.1	BF16	85.4	68.0	86.7	64.7	88.3	83.2	79.8	16.9	71.6	4.19
	Quantar	85.6	67.9	87.1	64.0	88.3	83.5	79.3	16.8	71.6	4.19
Qwen3-32B	BF16	82.4	63.9	84.4	57.8	86.1	80.8	73.2	26.6	69.4	10.21
	Quantar	82.5	64.1	84.1	58.4	85.6	80.8	72.5	26.6	69.3	10.25
Llama-3.1-8B	BF16	54.7	60.0	81.4	51.2	82.5	80.1	74.2	19.3	62.9	7.94
	Quantar	54.6	60.0	81.6	52.0	82.0	79.7	73.8	19.4	62.9	7.98

4 Evaluation

We evaluate Quantar on eight B300 GPUs with tensor parallelism of eight, E4M3 communication, and per-channel scales with headroom 32. Each checkpoint uses a single scale table, calibrated once and left unchanged across every task and context length. We compare three ways of communicating the partial outputs: BF16 All-Reduce through NCCL, the unmodified baseline; per-hop FP8, the default-ring path selected by NCCL's FP8 All-Reduce on our installation, which rounds the running sum at each stage; and Quantar. Quality is measured with lm-eval-harness (EleutherAI contributors, 2025) and RULER's own scorer against a live server, using the same prompts, seeds, and limits for every configuration. The evaluation covers MMLU-Pro (Wang et al., 2024b), WikiText-2 word perplexity (Merity et al., 2017), a zero-shot battery (Hendrycks et al., 2021; Zellers et al., 2019; Clark et al., 2018; 2019; Bisk et al., 2020; Sakaguchi et al., 2020; Paperno et al., 2016), RULER (Hsieh et al., 2024), LongBench-E (Bai et al., 2024), and GSM8K with the eight-shot chain-of-thought prompt from lm-eval (Cobbe et al., 2021; Wei et al., 2022). The calibration-transfer study additionally uses HumanEval pass@1 (Chen et al., 2021), the 11-language MGSM-direct mean (Shi et al., 2023), and Qasper-E from LongBench-E (Dasigi et al., 2021; Bai et al., 2024).

Table 1 compares Quantar with BF16 across dense, mixture-of-experts, and hybrid models from 8B to 671B parameters, with both BF16 and NVFP4 weights. Across the checkpoints, Quantar changes MMLU-Pro by at most 0.42 points and WikiText perplexity by at most 0.88%. On the sixth checkpoint, Nemotron-3-Ultra NVFP4, WikiText changes by -0.11%.

Table 2 extends the comparison to eight zero-shot tasks. On every model, the eight-task average is unchanged to within 0.1 points, and every task, together with all 33 MMLU-STEM and MMLU-Pro subtasks, lies within twice the BF16 arm's reported standard error (Section E).

Long contexts are where a frozen scale is most exposed, since one calibrated scale table must cover every sequence length. Table 3 shows that it does. On DeepSeek-V3.1, the RULER core average under Quantar remains within 0.1 points of BF16 at every length from 8k to 128k. On Llama-3.1-8B, the 13-task average remains within 0.5 points of BF16 through 64k. At 128k, the 11-task core differs by 0.9 points. Generation-heavy tasks show the same result (Table 4): the LongBench-E average

Table 3: RULER scores across context length. DeepSeek-V3.1 is reported by task category: needle retrieval (eight tasks), variable tracking, common- and frequent-word aggregation

Model	Tasks	Method	8k	16k	32k	64k	128k
DeepSeek-V3.1	Retrieval (8)	BF16	99.9	99.8	99.9	99.0	98.0
		Quantar	100.0	99.8	99.8	99.2	98.0
	Multi-hop (1)	BF16	100.0	100.0	100.0	100.0	100.0
		Quantar	100.0	100.0	100.0	100.0	100.0
	Aggregation (2)	BF16	98.7	99.0	96.6	90.3	86.2
		Quantar	98.6	99.2	96.7	90.2	85.9
	QA (2)	BF16	41.6	39.7	27.2	23.1	24.5
		Quantar	50.2	43.8	29.7	28.6	27.5
	Core avg. (11)	BF16	99.7	99.7	99.3	97.5	96.0
		Quantar	99.7	99.7	99.3	97.6	96.0
	All avg. (13)	BF16	90.8	90.4	88.2	86.1	85.0
		Quantar	92.1	91.1	88.6	87.0	85.4
Llama-3.1-8B	Average	BF16	92.3	91.3	88.9	80.8	73.8
		Quantar	92.4	91.3	88.4	80.7	74.7

Table 4: LongBench-E and eight-shot GSM8K-CoT on DeepSeek-V3.1 (scores $\times 100$). LongBench-E cells average two BF16 and three Quantar runs; the GSM8K-CoT cells are one BF16 run and the mean of two Quantar runs. Avg. is the macro average over the nine evaluated LongBench-E tasks.

	LongBench-E ↑					GSM8K-CoT ↑
Method	Single-doc QA	Multi-doc QA	Few-shot	Code	Avg.	strict match
BF16	12.4	9.2	39.9	33.5	25.6	87.3
Quantar	13.1	9.7	41.2	35.1	26.6	87.8

Table 5: Calibration transfer on DeepSeek-V3.1. The deployed WikiText-103 table is reused without refitting. HumanEval pass@1, MGSM-direct, and Qasper-E scores are multiplied by 100.

Scale table	WikiText-2 PPL ↓	HumanEval pass@1 ↑	MGSM-direct ↑	Qasper-E ↑
BF16	4.188	62.8	53.6	11.4
WikiText-103	4.193	64.6	53.7	9.9
WikiText-103	4.194	64.0	53.2	9.8
Code	4.190	64.6	53.0	9.6
Multilingual	4.196	62.2	53.5	8.8

differs by less than the spread between repeated runs of the same configuration, and GSM8K-CoT by less than one standard error.

Table 5 first keeps the deployed WikiText table frozen while moving to code and multilingual tasks: WikiText perplexity changes by 0.10%, HumanEval improves by 1.83 points, and MGSM changes by 0.11 points. It then changes the calibration source at equal budget. WikiText-, code-, and multilingual-calibrated tables span 0.0052 in WikiText perplexity and no more than their reported sampling uncertainty on the three downstream scores. Sampled telemetry observes no representability- guard engagements for any equal-budget table (Section B).

Table 6 measures why sharing matters on natural activations. Summing values with rank-local scales produces 41.3% median relative error because equal FP8 values no longer represent equal activation units. Reconstructing before the sum reduces this to 2.31%, but restores the conversion that direct reduction is meant to remove. Quantar performs the sum directly with 3.58% median error (Section C.4).

A simulated-rank sweep then recomputes real partials from the same captured activations and checkpoint weights. Quantar's median error remains 3.72% at $W = 2$, 3.16% at $W = 32$, and 3.52% at

Table 6: Relative L_2 error (%) over all 121 All-Reduce sites in a 256-row WikiText capture. Rank-local direct sum adds values expressed in different units; reconstruct-first converts them before summation.

Method	Median	P95	Maximum
BF16	0.36	0.38	0.42
Rank-local scales, direct sum	41.32	111.98	344.13
Rank-local scales, reconstruct first	2.31	2.83	3.41
Quantar	3.58	3.94	4.36

Table 7: *Top:* Batch-one B300 prefill latency (ms) at the two context lengths shared by all four checkpoints. *Bottom:* Model-level and complete communication-boundary speedup across interconnects.

	8k tokens			32k tokens		
Model	BF16	Quantar	Speedup	BF16	Quantar	Speedup
DeepSeek-V3.1	183.7	166.6	1.10×	591.6	492.3	1.20×
Llama-3.1-8B	36.8	28.6	1.28×	122.0	102.0	1.20×
Qwen3-32B	98.7	84.3	1.17×	401.1	368.1	1.09×
Mistral-Nemo-12B	51.1	43.1	1.18×	179.2	156.3	1.15×

Hardware	Scope	Model speedup	Communication speedup
B300 NVLink	Four checkpoints	1.09–1.28×	1.46–1.81×
RTX PRO 6000 PCIe	Qwen3-8B	1.39–1.61×	1.56–1.92×
	GPT-OSS-120B	1.23–1.53×	1.56–1.92×

$W = 64$, while per-hop FP8 grows from 3.72% to 15.24%. The $W = 16, 32, 64$ points are virtual repartitions, not hardware measurements (Section D).

Across four checkpoints, Quantar reduces prefill latency by 1.10–$1.28\times$ at 8k tokens and 1.09–$1.20\times$ at 32k. Quantar changes the wire precision, transport, and fused producer path together, so its speedup over NCCL BF16 is not a precision-only comparison. The matched transport comparison in Table 11 makes this clear. At 8k, BF16 over NVLS already gives $1.08\times$ forward speedup over NCCL BF16, while Quantar gives $1.10\times$, an additional $1.02\times$. At 32k, BF16 over NVLS gives $1.03\times$ and Quantar gives $1.20\times$, increasing the matched-NVLS gain to $1.17\times$. The collective itself is 1.46–$1.81\times$ faster from 2k to 32k; the full context-length, weight-precision, and transport sweeps are reported in Section F.

5 Limitations and Future Work

Quantar relies on normalization before the residual branch; post-norm architectures do not inherit the same request-independent ceiling without a separate argument. Sequence parallelism decomposes the same sum into reduce-scatter and all-gather, so the arithmetic carries over, but its end-to-end integration remains open. Expert, pipeline, and context parallelism use different communication patterns and have not been validated here. Training convergence remains outside the present analysis. Finally, weak alignment of deterministic rounding errors is a measured property rather than a universal guarantee.

6 Conclusion

Quantar shows that tensor-parallel All-Reduce activations can be quantized while preserving measured downstream quality and reducing prefill latency. Communication therefore need not remain the last BF16 step in an otherwise low-bit inference path. This becomes increasingly important as faster low-precision computation leaves communication as the next bottleneck.

AI DISCLOSURE

We used generative AI tools to help check the mathematical claims; refine hypotheses; organize the experiments, and draft the manuscript. We did not use these tools for translation, transcription, or qualitative data analysis. The authors ran all reported experiments and model evaluations, tested the code, verified numerical claims against preserved artifacts, manually reviewed the cited literature, and reviewed all AI-assisted content. We take responsibility for the final text, claims, code, figures, and experimental results.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *NeurIPS*, pp. 1710–1721, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6c340f25839e6acdc73414517203f5f0-Abstract.html.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300.

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, 2018.

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

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *NAACL*, pp. 4599–4610, 2021. doi: 10.18653/v1/2021.naacl-main.365.

DeepSeek-AI. DeepSeek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. DeepSeek-V3.1 model card. Hugging Face, 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-V3.1.

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

Harry Dong, Tyler Johnson, Minsik Cho, and Emad Soroush. Towards low-bit communication for tensor parallel LLM inference. In *Fourth Workshop on Efficient Natural Language and Speech Processing at NeurIPS*, 2024. URL https://arxiv.org/abs/2411.07942.

EleutherAI contributors. EleutherAI/lm-evaluation-harness: v0.4.9.2, 2025. URL https://zenodo.org/records/17728786.

FlashInfer contributors. flashinfer.comm.quantized_all_reduce — FP8 quantized two-shot AllReduce with per-block scaling. https://docs.flashinfer.ai/generated/flashinfer.comm.quantized_all_reduce.html, 2026.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.

Jan Hansen-Palmus, Michael Truong Le, Oliver Hausdörfer, and Alok Verma. Communication compression for tensor parallel LLM inference. In *Workshop on Machine Learning and Compression at NeurIPS*, 2024. URL https://openreview.net/forum?id=YkttwLldEb.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *COLM*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Aojie Jiang, Kang Zhu, Zhiheng Zhang, Zhengxu Su, Juntao Liu, Yuan Du, and Li Du. A switch-centric in-network architecture for accelerating LLM inference in shared-memory network, 2026. URL https://arxiv.org/abs/2603.28239.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3252–3261, 2019. URL https://proceedings.mlr.press/v97/karimireddy19a.html.

Haoyang Li, Wei Luo, Xinjun Niu, Spandan Tiwari, Ke Wang, Jiangyong Ren, Ashish Sirasao, and Doug Lehr. QuickReduce: Up to 3x faster all-reduce for vLLM and SGLang. https://rocm.blogs.amd.com/artificial-intelligence/quick-reduce/README.html, 2025a.

Qingyuan Li, Bo Zhang, Liang Ye, Yifan Zhang, Wei Wu, Yerui Sun, Lin Ma, and Yuchen Xie. Flash communication: Reducing tensor parallelization bottleneck for fast large language model inference, 2024. URL https://arxiv.org/abs/2412.04964.

Qingyuan Li, Bo Zhang, Hui Kang, Tianhao Xu, Yulei Qian, Yuchen Xie, and Lin Ma. Flash-Communication V2: Bit splitting and spike reserving for any bit communication, 2025b. URL https://arxiv.org/abs/2508.03760.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, volume 6, pp. 87–100, 2024.

Man Liu, Xingchen Liu, Xingjian Tian, Bing Lu, Shengkai Lyu, Shengquan Yin, Wenjing Huang, Zheng Wei, Hairui Zhao, Guangming Tan, and Dingwen Tao. TACO: Efficient communication compression of intermediate tensors for scalable tensor-parallel LLM training. In *HPDC*, pp. 235–248, 2026. doi: 10.1145/3806645.3807584.

Llama Team, AI at Meta. The Llama 3 herd of models, 2024.

matkle. [Feature] add Int8/FP8-quantized all-reduce. https://github.com/vllm-project/vllm/pull/39783, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 formats for deep learning, 2022.

NVIDIA. DeepSeek-V3.1-NVFP4 model card. Hugging Face, 2025a. URL https://huggingface.co/nvidia/DeepSeek-V3.1-NVFP4.

NVIDIA. NVFP4. Transformer Engine documentation, 2025b. URL https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/features/low_precision_training/nvfp4/nvfp4.html.

NVIDIA. NCCL user guide. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/, 2026a.

NVIDIA. Parallel thread execution ISA, version 9.4. https://docs.nvidia.com/cuda/parallel-thread-execution/, 2026b.

NVIDIA. NCCL device api: Remote reduce and copy. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/api/device_reducecopy.html, 2026c.

Open Compute Project. OCP microscaling formats (MX) specification, version 1.0. https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf, 2023.

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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144.

Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009. doi: 10.1016/j.jpdc.2008.09.002.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. FP8-LM: Training FP8 large language models, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Interspeech*, pp. 1058–1062, 2014. doi: 10.21437/Interspeech.2014-274.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *ICLR*, 2023. URL https://openreview.net/forum?id=fR3wGCk-IXp.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism, 2019.

Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 19 (1):49–66, 2005. doi: 10.1177/1094342005051521.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *NeurIPS*, pp. 14236–14245, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d9fbed9da256e344c1fa46bb46c34c5f0-Abstract.html.

Guanhua Wang, Heyang Qin, Sam Jacobs, Xiaoxia Wu, Connor Holmes, Zhewei Yao, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. ZeRO++: Extremely efficient collective communication for large model training. In *ICLR*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS Datasets and Benchmarks Track*, volume 37, pp. 95266–95290, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Miles Williams and Nikolaos Aletras. On the impact of calibration data in post-training quantization and pruning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10100–10118, 2024. doi: 10.18653/v1/2024.acl-long.544.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099, 2023.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. In *NeurIPS*, volume 35, pp. 27168–27183, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, pp. 12360–12371, 2019.

A Dynamic Quantization in Serving Systems

The serving systems cited in the Introduction choose communication scales from the tensor being sent. This section records the concrete paths behind that statement and separates scale selection from accumulation.

QuickReduce. QuickReduce computes one scale for each 32-value block on each rank during the current forward pass (Li et al., 2025a). Its exchanged INT4–INT8 blocks are reconstructed inside the collective before their contributions are accumulated. The scale is therefore dynamic and rank-local; the reduction remains meaningful because summation happens after reconstruction.

vLLM and FlashInfer. FlashInfer's quantized communication path and the implementation proposed in vLLM PR #39783 compute per-group scales from the current rank-local input (matkle, 2026; FlashInfer contributors, 2026). The vLLM PR was still under review on September 25, 2026. The receiver reconstructs the communicated blocks before or while accumulating them. Finer blocks better follow each input's local range, but every invocation still measures that range and carries its scale through the communication path.

Direct low-precision accumulation. NCCL also exposes FP8 All-Reduce (NVIDIA, 2026a). On our NCCL 2.28.9 installation, forcing FP8 onto NVLS is rejected while the default ring path runs and rounds its intermediate sums back to FP8. This removes per-rank reconstruction but introduces repeated low-precision rounding. These two designs are the alternatives summarized in Section 2.3: reconstruct rank-local units before summation, or sum directly and round the running result. Quantar changes the scale policy so that neither escape is required.

B Calibration and Range Control

The scale table contains one per-channel range for every covered collective. During a BF16 calibration pass, each rank records the largest magnitude of both its partial output and the completed sum. A cross-rank maximum then produces

$$m_{\ell,c} = \max_{t \in \mathcal{C}} \left\{ \max_r |x^{(r)}_{\ell,t,c}|, \left| \sum_r x^{(r)}_{\ell,t,c} \right| \right\}, \qquad s_{\ell,c} = \frac{32 m_{\ell,c}}{C}.$$

The first term covers what each rank sends; the second covers what the collective returns. The deployed DeepSeek table uses 266,240 WikiText-103 training tokens. The same vector is installed on every rank and remains unchanged after calibration.

What calibration data changes. The main transfer result (Table 5) separates two questions. First, one WikiText-103 table is reused on WikiText-2, HumanEval, MGSM-direct, and Qasper-E without refitting. Second, three tables are calibrated with the same 266,240-token budget on WikiText-103, Python code, or multilingual Wikipedia. Their downstream spreads remain within measured task uncertainty, and none of the sampled values engages the representability guard. For this model and budget, the corpus changes where calibration observes the range much less than it changes downstream quality.

Normalization intervention. The architectural argument predicts that changing residual-stream amplitude before RMSNorm should leave the branch input unchanged. Scaling after the normalization provides the direct control. Table 8 shows the same result across all three prompt domains. Scaling before RMSNorm leaves the communicated output at 0.999–$1.000\times$. Scaling the normalized output changes the branch input by exactly $4\times$, after which attention and SwiGLU produce their expected nonlinear responses. The intervention verifies amplitude suppression; the finite range still follows from the checkpoint-dependent bound in Section 3.

Table 8: RMSNorm intervention on five DeepSeek attention/MLP branches. Residual or normalized activations are multiplied by four on WikiText, code, and multilingual prompts. Entries are median per-channel absmax ratios against the unmodified run.

prompt	pre-RMSNorm increment	post-RMSNorm input	largest branch response
WikiText	1.000–1.000×	4.000×	30.2×
Code	1.000–1.000×	4.000×	34.4×
Multilingual	0.999–1.000×	4.000×	32.7×

C Reduction Arithmetic and Error

C.1 Implementation

We implement Quantar in SGLang at the row-parallel attention and MLP/MoE output boundaries. On B300, a custom NVLS CUDA kernel places each rank's E4M3 partial in symmetric memory, assigns one output shard to each rank, and uses `multimem.ld_reduce` with FP16 accumulation to form that shard inside NVSwitch. A `multimem.st` broadcasts the completed shards (NVIDIA, 2026b), after which a second kernel applies the shared scale once and fuses the residual addition and RMSNorm. The send-side quantizer limits each rank's value to C/W before writing the symmetric buffer, guaranteeing that the returned E4M3 sum is finite.

C.2 Representability of the Completed Sum

Sharing a scale makes every rank use the same unit, but the in-switch result must also fit in E4M3. Quantar divides the finite range among the W contributors:

$$\left|q_{\ell,c}^{(r)}\right| \le \frac{C}{W}.$$

The completed sum then satisfies

$$\left|\sum_{r=1}^{W} q_{\ell,c}^{(r)}\right| \le \sum_{r=1}^{W} \left|q_{\ell,c}^{(r)}\right| \le C.$$

This is a representability condition, independent of calibration and of any cancellation between ranks. Before the guard, aligned stress inputs reach 896, 1792, and 3584 E4M3 units at $W = 2, 4, 8$, producing a nonfinite return. The guarded path remains at or below 448. On measured WikiText traffic the guard changes zero of 1.18×10^{11} sampled values; a random-token stress run changes eight of 2.97×10^{10}. At 8k–32k, adding the guard changes measured boundary latency by at most 0.5%.

C.3 Accumulation Precision

Sharing a scale resolves the unit mismatch, but does not determine where a collective rounds its intermediate sums. Consider a binary reduction in which node v rounds the sum of its children back to FP8:

$$\mathbf{z}_v = \mathcal{R}_8(\mathbf{z}_{v_L} + \mathbf{z}_{v_R}) = \mathbf{z}_{v_L} + \mathbf{z}_{v_R} + \boldsymbol{\rho}_v.$$

The root then contains

$$\mathbf{z}_{\text{root}} = \sum_{r=1}^{W} \mathbf{q}_\ell^{(r)} + \sum_{v \text{ internal}} \boldsymbol{\rho}_v.$$

Applying the shared scale restores activation units but cannot remove the intermediate rounding terms.

Quantar instead widens the shared-scale FP8 operands inside the reduction, accumulates them with FP16 semantics, and converts only the completed sum back to FP8. This is not reconstruct-before-summation: the reduction applies no rank-specific scale and forms the sum directly in the common unit. The shared scale is applied once after the completed result is returned.

C.4 Why Rank-Local Scales Cannot Be Summed Directly

The two-rank collision in Section 3 is visible on natural activations at every covered site. In Table 6, directly summing values with rank-local scales gives 41.3% median relative error: the same quantized value denotes a different activation unit on each rank. Reconstructing each partial before summation reduces the median to 2.31%, but restores the conversion and full-width accumulation that direct reduction removes. Quantar performs the direct sum with one shared unit at 3.58%.

C.5 Worst-Case In-Range Rounding

E4M3 has three stored fraction bits. For a normal scaled value $|x|/s \in [2^e, 2^{e+1})$, adjacent representable values are separated by

$$\Delta_e = 2^{e-3}.$$

Round-to-nearest moves the value by at most half this spacing, so

$$|e(x;s)| \le s\frac{\Delta_e}{2} = s\,2^{e-4}.$$

Because $|x| \ge s\,2^e$ in the same exponent interval,

$$\frac{|e(x;s)|}{|x|} \le \frac{s\,2^{e-4}}{s\,2^e} = 2^{-4} = 6.25\%.$$

The scale cancels: within the normal range, increasing headroom changes which exponent interval a value occupies but not this worst-case relative bound. Subnormal values are covered separately by the lower-range discussion in Section 3.

C.6 Error Across Depth

The open-loop replay isolates the geometry of transport error. It captures 256 rows from all 121 DeepSeek sites, holds the branch outputs fixed, and replays their shared-scale E4M3 errors. The final coherence ratio is $\kappa_{121} = 1.114$, and the mean pairwise cosine is 8.10×10^{-4}. Relative to the accumulated unquantized branch-update stream, error peaks at 2.89% and ends at 1.31%; it does not grow monotonically.

The closed-loop capture asks a different question: what happens after later layers consume the perturbed residual stream? BF16 repeats are bitwise identical at all 121 sites, while Quantar's residual trajectory differs by 29.0–37.3% at its peak across WikiText, code, and multilingual prompts. This is an internal-state trajectory, not a quality metric. The served evaluations in Section 4 measure the resulting model outputs directly.

A controlled synthetic experiment verifies that the diagnostic distinguishes the two accumulation regimes: additive transport error follows exponent 0.500, while a deliberately fixed-sign control follows 0.992 over 128 residual updates. This control supports the interpretation of the real coherence measurement; it does not substitute for it.

D Scaling with Tensor-Parallel World Size

The hardware captures contain eight genuine rank partials. Grouping them gives exact virtual $W = 2, 4, 8$ inputs. To extend the same activation beyond TP8, we repartition the captured projection input and the checkpoint's actual dequantized weight, then recompute each virtual rank partial. The $W = 16, 32, 64$ points therefore preserve the activation and weights but are replay results, not hardware measurements.

Quantar remains nearly flat: 3.72% at $W = 2$, 3.16% at $W = 32$, and 3.52% at $W = 64$. Per-hop FP8 rises from 3.72% to 15.24% as it repeatedly rounds the running sum. At $W = 64$, world size first exceeds the shipped headroom 32. The representability guard then raises one MLP site's error to 7.78%; the other four installed-table sites remain between 3.15% and 3.88%. The five-site median therefore stays flat even as the guard exposes the limit of applying a TP8 table at TP64.

Table 9: Median relative L_2 over five DeepSeek sites with installed frozen scales. $W \leq 8$ is cross-checked against grouped TP8 partials; $W = 16, 32, 64$ uses virtual repartition.

transport	$W = 2$	$W = 4$	$W = 8$	$W = 16$	$W = 32$	$W = 64$
Quantar	3.72%	3.34%	3.21%	3.19%	3.16%	3.52%
per-hop FP8	3.72%	4.58%	5.74%	7.56%	10.97%	15.24%

E Quality Evaluation Details

All served quality comparisons use identical prompts, seeds, and task limits. MMLU-Pro uses all 12,032 questions. The zero-shot battery uses each task's full test set, and a task passes when its difference lies within twice the BF16 arm's reported standard error. LongBench-E averages two BF16 and three Quantar runs; GSM8K-CoT uses all 1,319 examples.

The main battery table reports the complete eight-task scores for DeepSeek, Qwen3, and Llama. All nine reported metrics and all 33 MMLU-STEM/MMLU-Pro subtasks satisfy the 2σ rule on each model. The full task matrix exposes both the sign and magnitude of every task-level change.

For RULER, Table 3 reports every DeepSeek category from 8k through 128k and every additional-model context completed by both arms. The 128k DeepSeek tier uses 100 examples per task; all shorter tiers use 500. Retrieval, multi-hop tracing (variable tracking), and aggregation remain within 0.3 points throughout. The larger QA movement comes from tasks where both arms frequently return an empty answer, which is why the paper reports categories rather than only the overall average.

LongBench-E and eight-shot GSM8K-CoT use generation-heavy outputs rather than multiple-choice scoring. The LongBench-E macro difference is smaller than the spread between repeated Quantar runs, and GSM8K-CoT differs by less than one standard error. Calibration transfer is reported separately in Section B because it changes the scale table rather than the downstream protocol.

F Latency Breakdown

The main paper reports the common 8k/32k model comparison. The full context sweep below retains absolute latency, while the following tables isolate transport, dtype, and serving concurrency.

At 8k, BF16 over the switch gives $1.09\times$ at the boundary and per-hop FP8 over NCCL gives $1.24\times$; Quantar combines the byte reduction with in-switch summation for $1.54\times$. At 32k, the same three values are $1.03\times$, $1.29\times$, and $1.46\times$. The corresponding Quantar forward gains are $1.10\times$ and $1.20\times$.

The concurrency sweep separates a shorter transport from a faster reduction. Shared FP8 over P2P and Quantar both reduce bytes, but only Quantar also uses the switch; per-hop FP8 repeatedly rounds and gives no serving gain.

On the PCIe-only RTX PRO 6000 node, model-level speedup reaches 1.39–$1.61\times$ on Qwen3-8B and 1.23–$1.53\times$ on GPT-OSS-120B; the complete communication boundary reaches 1.56–$1.92\times$. These are cross-platform performance results, not hardware twins of the B300 measurements.

Table 10: Batch-one prefill latency (ms) on eight B300 GPUs. The separate DeepSeek block reports the measured effect of FP8 versus NVFP4 weights over 8k–32k.

Model		8k	16k	32k	64k	128k
DeepSeek-V3.1	BF16 (ms)	183.7	280.8	591.6	1422.1	4153.1
	Quantar (ms)	166.6	233.8	492.3	1288.4	3664.9
	Speedup	1.10×	1.20×	1.20×	1.10×	1.13×
Llama-3.1-8B	BF16 (ms)	36.8	66.5	122.0	281.4	751.7
	Quantar (ms)	28.6	61.3	102.0	256.2	704.7
	Speedup	1.28×	1.08×	1.20×	1.10×	1.07×

DeepSeek-V3.1 speedup by weight precision	8k	16k	32k
FP8 weights	1.10×	1.09×	1.08×
NVFP4 weights	1.10×	1.16×	1.17×

Table 11: Transport and dtype attribution on DeepSeek-V3.1, reported as speedup over BF16 with NCCL. Collective columns measure the complete communication boundary; forward columns measure batch-one wall time.

		Collective		Forward	
Communication	Transport	8k	32k	8k	32k
BF16	NCCL	1.00×	1.00×	1.00×	1.00×
BF16	NVLS	1.09×	1.03×	1.08×	1.03×
Per-hop FP8	NCCL	1.24×	1.29×	1.00×	1.10×
Quantar (shared FP8)	NVLS	1.54×	1.46×	1.10×	1.20×

Table 12: Median-TTFT speedup over BF16 NCCL for random requests with 8,192 input tokens and one output token on eight B300 GPUs. Columns report client concurrency caps of 16, 64, and 128; each run issues four times as many requests as its cap. Each cell is the ratio of arm medians over three repetitions.

communication	cap 16	cap 64	cap 128
Quantar	1.071×	1.119×	1.103×
shared FP8 (P2P)	1.082×	1.114×	1.090×
per-hop FP8	0.974×	0.993×	0.996×

G Experimental Protocol and Reproducibility

The primary system contains eight NVIDIA B300 GPUs connected by NVLink/NVSwitch. Preserved runs record PyTorch 2.11.0+cu130, CUDA 13.0.88, driver 580.178.04, Python 3.12.14, NCCL 2.28.9, and lm-eval-harness 0.4.9.2. The RTX comparison uses eight RTX PRO 6000 GPUs connected through PCIe across two NUMA nodes.

Quality runs use seed 1234 and identical prompts and limits in both arms. Forward timing interleaves configurations, drops the cold pass, and reports the median of at least three repetitions. Collective measurements use CUDA events after numerical validation. Serving alternates modes in one server, flushes the cache, warms after each change, and reports arm-median TTFT. Every accepted run checks finite outputs.

Served lm-eval, RULER, LongBench-E, and GSM8K establish downstream quality. Collective, forward, and serving measurements establish hardware performance. Ownership, coherence, closed-loop, and world-size results replay or capture real activations; the $W = 16, 32, 64$ points are virtual repartitions. Synthetic inputs appear only in controlled numerical diagnostics.