Towards Low-bit Communication for Tensor Parallel LLM Inference

Harry Dong[*]
Carnegie Mellon University
harryd@andrew.cmu.edu

Tyler Johnson
Apple

Minsik Cho
Apple

Emad Soroush
Apple

Abstract

Tensor parallelism provides an effective way to increase server large language model (LLM) inference efficiency despite adding an additional communication cost. However, as server LLMs continue to scale in size, they will need to be distributed across more devices, magnifying the communication cost. One way to approach this problem is with quantization, but current methods for LLMs tend to avoid quantizing the features that tensor parallelism needs to communicate. Taking advantage of consistent outliers in communicated features, we introduce a quantization method that reduces communicated values on average from 16 bits to 4.2 bits while preserving nearly all of the original performance. For instance, our method maintains around 98.0% and 99.5% of Gemma 2 27B's and Llama 2 13B's original performance, respectively, averaged across all tasks we evaluated on.

1 Introduction

The use of large language models (LLMs) [6, 8, 20, 21] has ballooned in countless areas due to their impressive capabilities. Even so, with the enormous size of server LLMs, inference-time efficiency becomes a dire issue for those who own the models and those who use them. Fortunately, techniques like sequence parallelism [9] and tensor parallelism [14, 19] distribute the computational load of transformer-based LLMs [23] onto different devices. However, these methods require communication between devices, which is especially a concern when serving models, so cheaper networking would greatly cut costs. In addition, as LLMs increase in size, we need to distribute them over more devices, which further drives up communication costs.

A natural idea is to try quantization, but this comes with challenges. Quantization methods for LLMs have largely focused on weights [4, 7, 10, 15, 18] or multiplication between low-bit weights and low-bit input features [1, 5, 24, 25, 27]. These methods are most useful for hardware-constrained settings, but their savings are not as relevant to tensor parallel server LLMs. In particular, current LLM quantization methods keep the output features of each attention and feedforward block at high precision (BF16 or FP16) to preserve performance. However, this is exactly what needs to be communicated in tensor parallelism. There has been work in quantized communication for distributed inference [12, 13, 16], but applications in LLMs have been limited. *Consequently, the main challenge with quantization is to find a way to communicate low-bit output features from tensor parallelized attention and feedforward blocks across devices while preserving the original model's performance.*

Thankfully, there are a couple observations that we can leverage. First, the communicated features have consistent structures. Looking at the aggregated quantization ranges for each feature across a calibration set in Figure 1 (details in Section 3), we observe that a small number of features have enormous ranges, potentially resulting in large quantization errors. Second, tensor parallelism can counteract feature quantization error. Theoretically, instead of uniformly distributed quantization

[*]Work done at Apple.

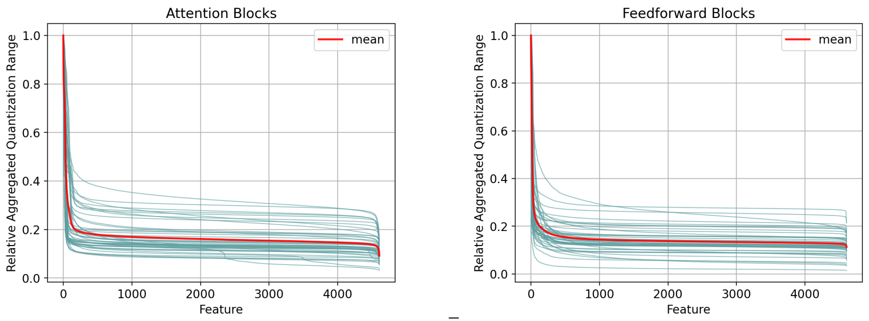

Figure 1: Sorted aggregated quantization ranges, $\bar{\boldsymbol{R}}_j$, of each each attention (left) and feedforward (right) block in Gemma 2 27B, with the mean across all layers in red. Values are scaled such that the max range for each layer is set to 1.

errors, quantizing the partial sums prior to synchronization results in aggregated errors that follow the Irwin-Hall distribution, approximately Gaussian as the number of devices increases. *This means tensor parallelism synchronization pushes quantization errors to be clustered around 0.* We take advantage of both observations in our method design.

We propose a solution to reduce tensor parallelism communication cost from 16 bits to less than 4.2 bits per value with minimal performance degradation. Inspired by the extra care taken for outlier features in many LLM quantization methods, the main idea of our algorithm is to determine a static set of features that are kept in BF16 while quantizing everything else to 4 bits without perturbing the weights. In Section 2, we formulate the problem with compressed communication for tensor parallelism. Next, Section 3 illustrates our method to select features to maintain at higher precision and how they are used during inference. Then, Section 4 showcases the performance of our method across multiple LLMs and tasks. For example, our method preserves around 98% of Gemma 2 27B's original performance despite communicating only about 1/4 of the information.

2 Tensor Parallelism Synchronization

Here, we formalize the communication problem in tensor parallelism that we aim to tackle. In tensor parallelism, weights in attention and feedforward blocks are partitioned across devices such that each partition can be computed in parallel until a synchronization step aggregates all outputs together on each device. Attention blocks can be split across the head dimension, and feedforward blocks can be split across the intermediate feature dimension. Synchronization in the form of an AllReduce is necessary after the output projection in attention blocks and after the down projection in feedforward blocks. With this setup, a tensor parallelized model and the original model produce the same outputs.

Define the final linear layer (i.e., the layer immediately before synchronization) in attention blocks or feedforward blocks as $f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$ for $\boldsymbol{x} \in \mathbb{R}^{1\times D}$, weight $\boldsymbol{W} \in \mathbb{R}^{D\times E}$, and bias $\boldsymbol{b} \in \mathbb{R}^{1\times E}$ when on a single device. When tensor parallelized, the input and weight take the form $\boldsymbol{x}^{(i)} \in \mathbb{R}^{1\times \frac{D}{N}}$ and $\boldsymbol{W}^{(i)} \in \mathbb{R}^{\frac{D}{N}\times E}$, respectively, for N devices. The pre-sync point linear layer on each device is:

$$f^{(i)}(\boldsymbol{x}^{(i)}) = \boldsymbol{x}^{(i)}\boldsymbol{W}^{(i)},$$

and a sum of the outputs from all devices produces the same activations as the original layer:

$$f(\boldsymbol{x}) = \left[\sum_{i=1}^{N} f^{(i)}(\boldsymbol{x}^{(i)})\right] + \boldsymbol{b}.$$

This operation requires communicating $f^{(i)}(\boldsymbol{x}^{(i)}) \in \mathbb{R}^E$ for each device, which can be expensive for large batch sizes and long sequences. Therefore, we aim to find compression and decompression functions, $\mathcal{C}^{(i)}$ and $\mathcal{D}^{(i)}$, so that we can communicate just $\mathcal{C}^{(i)}(f^{(i)}(\boldsymbol{x}^{(i)}))$ between devices and decompress as needed. Good functions should satisfy for all i and $\boldsymbol{x}^{(i)}$:

$$f^{(i)}(\boldsymbol{x}^{(i)}) \approx \mathcal{D}^{(i)}(\mathcal{C}^{(i)}(f^{(i)}(\boldsymbol{x}^{(i)}))).$$

We choose $\mathcal{C}^{(i)}$ and $\mathcal{D}^{(i)}$ to be quantization and dequantization operations based on the intuition that tensor parallelism can alleviate some of the quantization error in low-bit features.

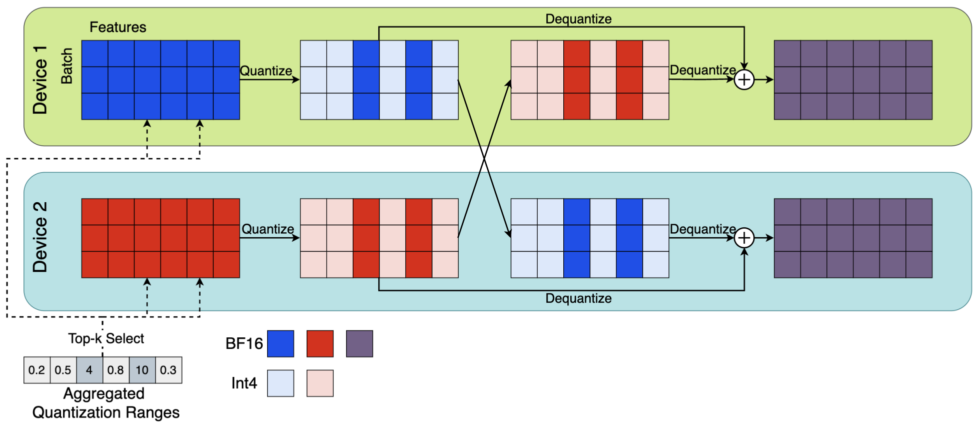

Figure 2: Our hybrid quantization algorithm. A small set of features are selected based on aggregated quantization ranges to be kept at BF16 while all others are quantized to Int4 prior to inter-device communication. Then, all tensors are converted to BF16 and summed to sync across each device.

3 Method

Outlined in Figure 2, we choose to transfer a fraction of features in BF16 and the rest in 4-bit precision. The BF16 features are chosen based on the range from the calibration set since a larger range would result in greater quantization error.

Calibration. We let each feature on each device have static and independent quantization parameters (i.e., NE sets of quantization parameters per tensor parallel block). These parameters are exponential moving averages of the minimum and maximum values that are obtained for each feature on each device on a calibration set. In other words, we find quantization parameters $(\boldsymbol{m}_j^{(i)}, \boldsymbol{M}_j^{(i)})$ for $1 \leq i \leq N$ and $1 \leq j \leq E$ following the recursive update rules:

$$\boldsymbol{m}_j^{(i)} = (1-\gamma)\boldsymbol{m}_j^{(i)} + \gamma \min(\boldsymbol{Y}_{\cdot,j}^{(i)}) \tag{1}$$

$$\boldsymbol{M}_j^{(i)} = (1-\gamma)\boldsymbol{M}_j^{(i)} + \gamma \max(\boldsymbol{Y}_{\cdot,j}^{(i)}) \tag{2}$$

for a sequence's features to communicate $\boldsymbol{Y}^{(i)} = f^{(i)}(\boldsymbol{X}^{(i)}) \in \mathbb{R}^{S \times E}$ and constant γ. For the first sequence, $\boldsymbol{m}_j^{(i)} = \min(\boldsymbol{Y}_{\cdot,j}^{(i)})$ and $\boldsymbol{M}_j^{(i)} = \max(\boldsymbol{Y}_{\cdot,j}^{(i)})$. Using symmetric quantization, the range on each device is $\boldsymbol{R}_j^{(i)} = 2\max(-\boldsymbol{m}_j^{(i)}, \boldsymbol{M}_j^{(i)})$. For our experiments, we use 256 random WikiText [11] sequences for calibration following update rules (1) and (2) with $\gamma = 0.01$.

Selecting High Precision Features. Looking at the aggregated quantization ranges across devices after calibration, $\bar{\boldsymbol{R}}_j \coloneqq \sum_{i=1}^{N} \boldsymbol{R}_j^{(i)}$, in Figure 1, we see a problem: a small set of features have wide ranges which harms the quantization quality. As a solution, we select the top-k features based on $\bar{\boldsymbol{R}}_j$ to be communicated at higher precision, fixed across all sequences and devices. All other features are symmetrically quantized to Int4. Compared to plain quantization, the additional overhead of our method includes the pre-sync BF16 feature selection and post-sync concatenation of features.

4 Experiments

Evaluating on multiple LLMs and tasks, we demonstrate that our quantization method preserves nearly all of the original performance at less than 4.2 bits per value. Furthermore, we show that even at lower and higher precision quantization, we still outperform all our baselines.

Our two baselines and our method use the same symmetric quantization parameters per model. For the first baseline, we quantize everything to Int4. For the second baseline, we randomly select k features with uniform probability to be kept at BF16, and everything else is quantized to Int4. For all experiments, our method and the second baseline fix $k = \lfloor E/64 \rfloor$ which brings the average bits per value to under 4.2 for both. The choice of $1/64$ as the fraction of BF16 features can be substituted,

Table 1: Zero-shot accuracy. The best values for each model and task, excluding the performance of full communication, are in bold. We fix $k = \lfloor E/64 \rfloor$ for Random BF16 and Selected BF16 (ours).

Method	Bits	ARC-e	ARC-c	WinoGrande	HellaSwag	BoolQ
Gemma 2 27B	16	87.71	62.37	79.08	65.38	84.67
Int4	4	84.13	57.51	74.27	63.50	83.55
Int4 + Random BF16	4.2	83.71	56.83	72.61	63.58	82.14
Int4 + Selected BF16	4.2	**86.45**	**61.01**	**76.48**	**63.86**	**83.98**
Llama 2 13B	16	79.55	48.72	72.22	60.03	80.58
Int4	4	77.40	47.10	68.67	58.88	78.01
Int4 + Random BF16	4.2	76.73	46.08	67.80	58.95	77.55
Int4 + Selected BF16	4.2	**79.08**	**47.35**	**72.93**	**59.49**	**81.22**
Mistral NeMo 12B	16	82.91	55.80	73.01	62.82	85.17
Int4	4	79.50	50.77	71.43	60.97	82.84
Int4 + Random BF16	4.2	79.12	51.02	70.56	60.96	81.04
Int4 + Selected BF16	4.2	**81.06**	**52.05**	**71.98**	**61.74**	**83.06**

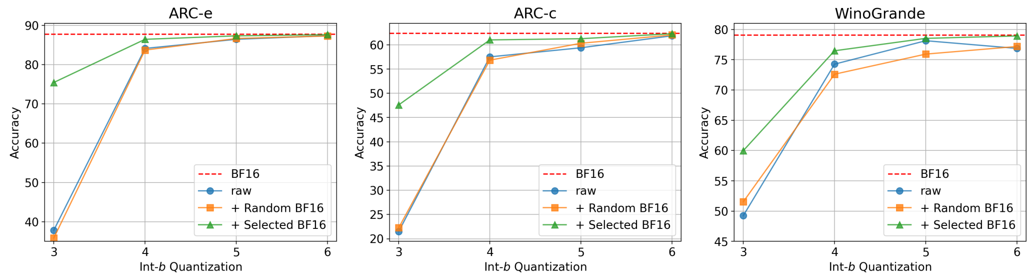

Figure 3: Gemma 2 27B performance when features are quantized to varying numbers of bits. Our method achieves the best accuracy for every quantization precision.

but we found this fraction to strike a good balance between performance and compression (Figure 1 suggests k can be relatively small). Future work can explore adaptively varying k per layer or input.

Using Gemma 2 27B [20], Llama 2 13B [22], and Mistral NeMo 12B [21], we compare our method against the baselines and full models on ARC-easy/challenge [3], WinoGrande [17], HellaSwag [26], and BoolQ [2]. Experimental results are reported for tensor parallelism across 8 devices. From Table 1, we see that our method achieves the best performance for the vast majority of tasks and models among the quantization schemes. Moreover, performance degradation with our method is fairly small in most cases. Overall, our method preserves around 98.0%, 99.5%, and 97.1% of the original Gemma 2 27B, Llama 2 13B, and Mistral NeMo 12B performance in Table 1, respectively. We also observe that choosing random features to be sent in BF16 adds virtually no benefit on top of pure Int4 quantization while choosing features based on aggregated quantization ranges adds a clear performance boost. In fact, in some cases, random selection appears to degrade performance in comparison to pure Int4 quantization, suggesting that maintaining low magnitude features at high precision without preserving high magnitude ones to be harmful. Beyond Int4 quantization, our method consistently best preserves performance at lower and higher precision, as seen in Figure 3.

5 Conclusion & Future Work

We introduced a method to quantize features for compressed synchronization in tensor parallel LLMs with very little performance degradation. Directly inspired by the consistent nature of outliers in these communicated features, our method combines Int4 and BF16 representations to compress these features to less than 4.2 bits per value. Nevertheless, there are many possibilities for future work. First, we would like to develop a system level implementation of our method to better assess the efficiency gains. Second, our method is fit for AllReduce executed as an AllGather followed by a local reduction, so it would be interesting to see how we can adapt our method to other AllReduce algorithms (e.g. ring-AllReduce). Together, our work and these directions would greatly improve server LLM inference efficiency.

References

[1] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

[2] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[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:1803.05457v1*, 2018.

[4] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR, 2023.

[5] 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*, 35:30318–30332, 2022.

[6] 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.

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

[8] Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

[9] Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. Sequence parallelism: Making 4d parallelism possible. *arXiv preprint arXiv:2105.13120*, 2021.

[10] 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. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[12] Aritra Mitra, John A Richards, Saurabh Bagchi, and Shreyas Sundaram. Distributed inference with sparse and quantized communication. *IEEE Transactions on Signal Processing*, 69: 3906–3921, 2021.

[13] V Sriram Siddhardh Nadendla, Yunghsiang S Han, and Pramod K Varshney. Distributed inference with m-ary quantized data in the presence of byzantine attacks. *IEEE transactions on signal processing*, 62(10):2681–2695, 2014.

[14] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[15] Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

[16] Minghai Qin, Chao Sun, Jaco Hofmann, and Dejan Vucinic. Disco: Distributed inference with sparse communications. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2432–2440, 2024.

[17] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[18] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[19] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[20] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[21] Mistral AI Team. Mistral nemo, 2024. URL `https://mistral.ai/news/mistral-nemo/`.

[22] 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. *arXiv preprint arXiv:2307.09288*, 2023.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. advances in neural information processing systems. *Advances in neural information processing systems*, 30(2017), 2017.

[24] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414, 2022.

[25] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[26] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[27] Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. Lqer: Low-rank quantization error reconstruction for llms. *arXiv preprint arXiv:2402.02446*, 2024.