

FAST: A Scalable Framework for Accelerating Flexible Structured Sparse Training

Shuaiheng Li^{1*}, Jun Liu^{1*}, Xinhao Li⁴, Yaoxiu Lian¹, Tianlang Zhao¹, Li Ding¹, Guohao Dai^{1,2,3†}

¹Shanghai Jiao Tong University, ²Infinigence-AI, ³SII, ⁴Southeast University

*Equal contributions, †Corresponding author: daiguohao@sjtu.edu.cn

Abstract—Sparse training is a critical approach to reducing the storage requirement while maintaining the model’s ability. However, it is non-trivial to apply the flexible structured sparsity (flex-SS) patterns during sparse training, which achieves Pareto optimality in terms of hardware efficiency and flexibility. We propose FAST, a fast and scalable framework that supports LLM training with flex-SS patterns. First, we propose a probability-based decoupling method that eliminates dependencies between tiles to generate the flex-SS mask efficiently. Second, we propose a weight-distribution-aware pivot search strategy that narrows down the available region of pivot candidates to reduce the communication overhead. Extensive experimental results show that FAST achieves up to 10.40× and 1.56× end-to-end training speedup compared with PyTorch and the SOTA framework.

Index Terms—Large Language Model, Sparse Training System.

I. INTRODUCTION

Large language models (LLMs) demonstrate their strong ability in all kinds of tasks. However, their incredible abilities come at the cost of innumerable storage resources. Sparsification methods are key techniques for reducing computation and storage demands, and according to the prior works [1], In-training sparsification is usually better than post-training sparsification. Existing works on sparse training mainly focus on 2:4 sparsity pattern, which is well supported by NVIDIA GPU. Despite flex-SS [2] lies on the Pareto frontier in hardware efficiency and flexibility, it is not trivial to apply flex-SS pattern during sparse training, due to the following two challenges: 1) The mask generation for the flex-SS patterns with target sparsity under-utilizes the GPU. 2) the pivot search process in the mask generation induces large communication overhead with model parallelism, due to the model parallelism partitions weights across different devices.

We propose FAST, an efficient and scalable framework for accelerating sparse training of LLMs, shown in Fig. 1. For *Challenge-1*, we decouple the sparsity configuration selection process among tiles, unlocking the parallel processing abilities of GPUs. For *Challenge-2*, we reduce the search space of the pivot according to the weight distribution, reducing the communication overhead under model parallelism. Our optimization method is orthogonal to existing sparse training work, and can be used to scale their methods to a larger scale.

We validate the performance of FAST in several LLMs and configurations. Experimental results show that FAST achieves 10.40× and 1.56× end-to-end training speedup compared with PyTorch and the SOTA framework.

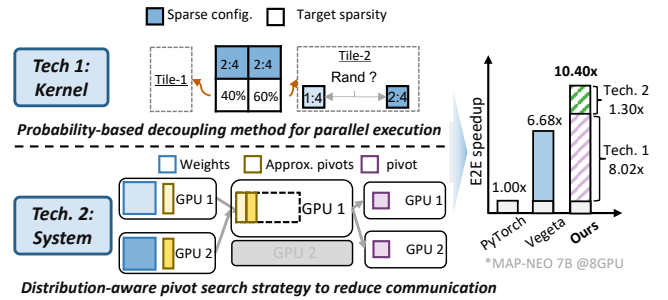


Fig. 1. Overview of FAST, an efficient and scalable sparse training framework.

II. PROBABILITY-BASED DECOUPLING METHOD FOR MASK GENERATION

Our core contribution lies in employing probabilistic selection to decouple global sparsity requirements from fixed tile-wise configurations. As illustrated in Fig. 2(a), the optimal sparse configuration for each tile is determined concurrently through a three-step process. First, we identify two candidate configurations that bracket the target tile-wise sparsity, filtering out all other unsuitable ones. Second, we calculate a selection probability based on these candidates to ensure the expected sparsity aligns with the target sparsity. Finally, the specific configuration is determined by a random number uniformly distributed in the interval $[0, 1]$. By leveraging the Law of Large Numbers across numerous tiles within the weight matrix, the final sparsity converges to the target sparsity.

We implement a CUDA kernel to leverage the parallel computing capabilities of GPUs and employ a heuristic mechanism that determines the optimal mapping of tiles to either threads or thread blocks.

III. DISTRIBUTION-AWARE PIVOT SEARCH STRATEGY

The pivot serves as the threshold for determining weight importance. Our search strategy narrows the possible space for Top-K selection by leveraging the Gaussian distribution ($\mu \approx 0$) in model’s weights. Shown in Fig. 2(b), we first estimate the Gaussian distribution of the entire weights through minimal statistic data. Then, we can analytically estimate weight density and apply a quantile function to create multiple equal-frequency bins. Finally, a lightweight all-reduce operation is performed on these bin counts. This balanced partitioning allows the algorithm to reduce the “pivot” weight’s interval by a factor

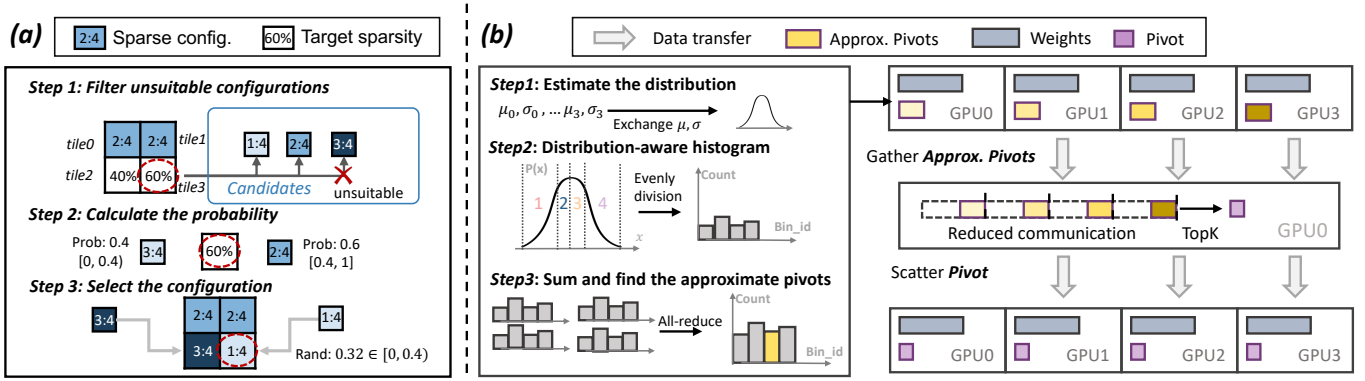


Fig. 2. Details of FAST. (a) the workflow of the decoupling method. (b) the workflow of the pivot search strategy.

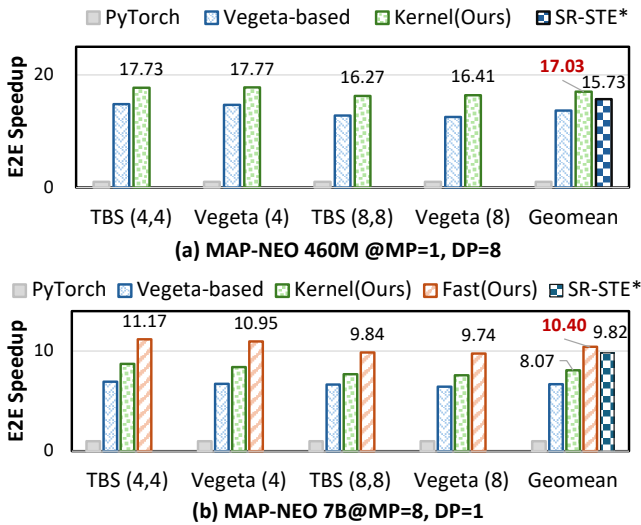


Fig. 3. The speedup of MAP-NEO 460M and Llama-2 13B on a single node with 8 A800 GPUs. TBS(x,x) denotes a flex-SS pattern with tile shape (x,x), Vegeta(x) denotes a flex-SS pattern with tile shape (1, N), with the minimum pruning unit being x. SR-STE uses 2:4 sparse pattern

proportional to the number of bins and significantly lowers the subsequent communication and sorting overhead.

IV. EVALUATION

We compare the performance of FAST with the Vegeta-based [3] method, and the SR-STE [1], while using the MAP-NEO 460M and MAP-NEO 7B. According to Fig. 3, Our kernel achieves an average $17.03 \times / 1.25 \times$ speedup on MAP-NEO 460M compared with the PyTorch implementation and the Vegeta-based method, and our FAST framework achieves an average $10.40 \times / 1.56 \times$ speedup on MAP-NEO 7B compared with the PyTorch implementation and the Vegeta-based method under model parallelism (MP). Meanwhile, compared with the 2:4 sparsity training, which is suitable for GPU training, we still have some performance advantages.

We evaluate the correctness of our framework by training the real-world sparse LLM. In order to make fair comparisons with dense models, we scale the sparse models and keep the number of non-zero weights the same. As Table I shows, the TBS(8:8)

TABLE I
EVALUATION RESULTS ACROSS DIFFERENT SPARSITY PATTERNS.

Dataset	Dense (official)	Dense	4:8*	US*	TBS(8:8)*
PIQA	0.68	0.66	0.67	0.67	0.69
ARC-e	0.49	0.58	0.61	0.61	0.63
ARC-c	0.22	0.25	0.26	0.25	0.28
Lambada	0.38	0.31	0.33	0.33	0.33
Geomean	0.41	0.42	0.43	0.43	0.45

stands out from the other patterns in the accuracy test, achieving first place in PIQA and ARC-e/c, and the other sparse models are also comparable with the dense model. The phenomenon is consistent with previous works on sparse training. Meanwhile, our dense model trained with 370B tokens has similar final results to the official dense model trained on 1T tokens, which proves our training recipe is acceptable.

V. CONCLUSION

We proposed FAST, which contains a novel probability-based decoupling method and a distribution-aware search strategy to reduce the mask generation overhead. Experiment shows that FAST achieves $10.40 \times$ and $1.56 \times$ end-to-end training speedup compared with PyTorch and the SOTA framework, and even exceeds the 2:4 sparse training.

VI. ACKNOWLEDGMENT

This work was sponsored by the National Natural Science Foundation of China (No. 62561160156) and Shanghai Rising-Star Program (No. 24QB2706200).

REFERENCES

- [1] A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun, and H. Li, "Learning n: m fine-grained structured sparse neural networks from scratch," *arXiv preprint arXiv:2102.04010*, 2021.
- [2] J. Liu, S. Zeng, J. Zhao, L. Ding, Z. Wang, J. Li, Z. Zhu, X. Ning, C. Zhang, Y. Wang *et al.*, "Tb-stc: Transposable block-wise n: M structured sparse tensor core," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 949–962.
- [3] G. Jeong, S. Damani, A. R. Bambhaniya, E. Qin, C. J. Hughes, S. Subramoney, H. Kim, and T. Krishna, "Vegeta: Vertically-integrated extensions for sparse/dense gemm tile acceleration on cpus," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 259–272.