Accelerating Transformer on GPUs with Sparse Ternary Weight Matrix

Yushi Ogiwara
yushiogiwara@keio.jp
Keio University

Hideyuki Kawashima
river@sfc.keio.ac.jp
Keio University

Abstract

Transformer architecture's computational cost increases as the number of parameters increases. Quantization methods with ternary weight matrices have addressed this issue. While existing approaches aim to utilize the property of ternary matrices on FPGA or special hardware, this paper proposes an efficient calculation method on GPUs when ternary matrices are sparse.

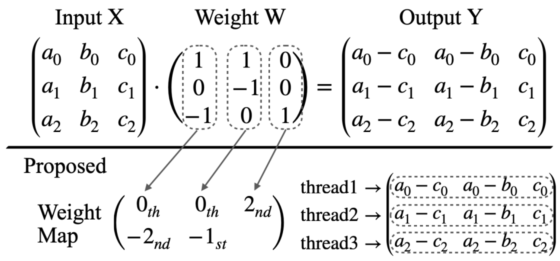

Figure 1: Multiplication with a ternary matrix (upper part). We propose a weight map for managing non-zero elements in the W matrix (lower part).

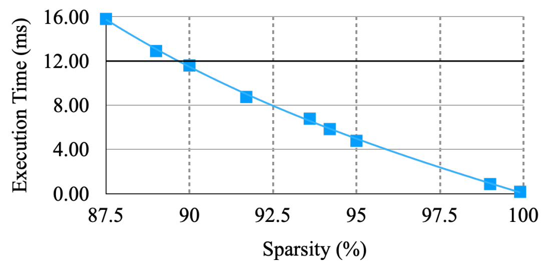

Figure 2: Execution time of multiplication of X matrix (4090×4096) and ternary W matrix (4096×16384).

1 Background

Transformer architecture [5] is scalable, but it is computationally expensive [1]. Quantization methods using ternary matrix [3, 6, 7], where each element is composed of $\{-1, 0, 1\}$, for weight matrix collects attention. This is because multiplication with a ternary matrix is efficient as only addition and subtraction are required, as shown in Fig. 1. This characteristic can be exploited for more acceleration of the Transformer.

2 Proposal

Existing approaches accelerated ternary weight matrix quantization on FPGA or special hardware [2, 3, 7]. We believe this method can also be leveraged with GPUs. We propose managing $\{-1, 1\}$ in the W matrix as a *weight map*, as shown in Fig. 1. Elements in the weight map indicate the index of 1 and -1 element in W per column where a sign is equipped on -1. CUDA threads read this map and calculate the rows of the Y matrix. As we can entirely skip the computation of zero elements in the W matrix, this method can perform better than merely using Tensor Core [4].

We compared the execution time of multiplying 4096×4096 matrix X and 4096×16384 ternary matrix W using Tensor Core and our proposed method on NVIDIA H100.

The proposed method performs better than the method using Tensor Core when more than about 90 % sparsity as shown in Fig. 2. This acceleration is due to skipping the calculation of zero elements in the W matrix.

Acknowledgments

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017) and the project (JPNP16007) commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and JSPS KAKENHI Grant Number 22H03596 and 24KJ1966.

References

[1] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022.
[2] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.
[3] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024.
[4] NVIDIA. Nvidia a100 tensor core gpu architecture, 2020.
[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.
[6] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu. Ternarybert: Distillation-aware ultra-low bit bert, 2020.
[7] R.-J. Zhu, Y. Zhang, E. Sifferman, T. Sheaves, Y. Wang, D. Richmond, P. Zhou, and J. K. Eshraghian. Scalable matmul-free language modeling, 2024.