OPS: Outlier-aware Precision-Slice Framework for LLM Acceleration

Fangxin Liu[1,2†], Ning Yang[1,2†], Zongwu Wang[1,2], Xuanpeng Zhu[3], Haidong Yao[3], Xiankui Xiong[3], Qi Sun[4], Li Jiang[1,2]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Shanghai Qi Zhi Institute, [3]ZTE Corporation, China, [4]Zhejiang University
{yn937391832, liufangxin, ljiang_cs}@sjtu.edu.cn

***Abstract*—Large language models (LLMs) have transformed numerous AI applications, with on-device deployment becoming increasingly important for reducing cloud computing costs and protecting user privacy. However, the astronomical model size and limited hardware resources pose significant deployment challenges. Model quantization is a promising approach to mitigate this gap, but the presence of outliers in LLMs reduces its effectiveness. Previous efforts addressed this issue by employing compression-based encoding for mixed-precision quantization. These approaches struggle to balance model accuracy with hardware efficiency due to their value-wise outlier granularity and complex encoding/decoding hardware logic. To address this, we propose OPS (Outlier-aware Precision-Slicing), an acceleration framework that exploits massive sparsity in the higher-order part of LLMs by splitting 16-bit values into a 4-bit/12-bit format. Crucially, OPS introduces an early bird mechanism that leverages the high-order 4-bit computation to predict the importance of the full calculation result. This mechanism enables efficient computational skips by continuing execution only for important computations and using preset values for less significant ones. This scheme can be efficiently integrated with existing hardware accelerators like systolic arrays without complex encoding/decoding. As a result, OPS outperforms state-of-the-art outlier-aware accelerators, achieving a $1.3-4.3\times$ performance boost with minimal model accuracy loss. This approach enables more efficient on-device LLM deployment, effectively balancing computational efficiency and model accuracy.**

I. Introduction

Transformers contain non-ignorable outliers that require high bit-widths to maintain accuracy, leading to significant computational and storage redundancy [1], [2]. Existing compression-based approaches attempt to reduce this redundancy by isolating or handling outliers [4], [5]. However, these methods introduce serialization and deserialization overheads that offset compression benefits and increase computational or storage burdens [3].

To this end, we propose an outlier-aware acceleration scheme that avoids complex overhead. Our approach quantizes weights and activations into 16-bit integers, ensuring comprehensive support for a wide range of model sizes and architectures. As illustrated in Figure 1, slicing the quantized values reveals considerable overall sparsity, even for unfavorably quantized activation value weights. After thorough analysis, we opted for a 4-12 bit partitioning paradigm: 4 bits for the higher-order part and 12 bits for the lower-order part.

† Both authors contributed equally to the paper. This work is supported by the National Natural Science Foundation of China (Grant No.62402311) and Natural Science Foundation of Shanghai (Grant No.24ZR1433700). Li Jiang is the corresponding author.

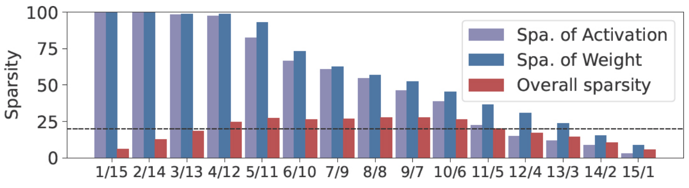

Fig. 1: Activation, Weight and overall sparsity of high-order part with different precision-slice method.

Such a partitioning offers several advantages. The 4-bit higher-order part exhibits over 95% sparsity, even for activations, opening new avenues for efficient quantization and compression. It also helps identify important outliers, enabling selective computation skipping for faster processing. The 12-bit lower-order part can be divided into three 4-bit segments, naturally aligning with memory and hardware, eliminating the need for complex alignment.

Based on these insights, we introduce OPS (Outlier-aware Precision-Slice), a framework that treats quantized values as two independent parts—high-precision and low-precision—processed in parallel through precision-slicing. OPS splits 16-bit quantized model parameters into 4-bit and 12-bit segments, both participating in parallel computation in 4-bit format. Through interleaved computation and simple comparators, OPS leverages the sparsity of higher-order parts to achieve energy savings and performance acceleration.

II. OPS ALGORITHM

Based on the $4/12$ bit division, we propose a precision-slicing computation flow with an **Early Bird** mechanism. For 16-bit numbers A and B, we split and execute multiplication in 4-bit format. The early bird mechanism leverages the partial sum from the high-order 4-bit computation to predict the importance of the full calculation. We first calculate the product of their higher-order parts and perform a threshold detection. If this partial result exceeds a predetermined threshold, we consider the calculation "significant" to the neural network and proceed to complete the full computation. Conversely, we deem it less important, mask the result to a preset value, and skip the remaining calculations.

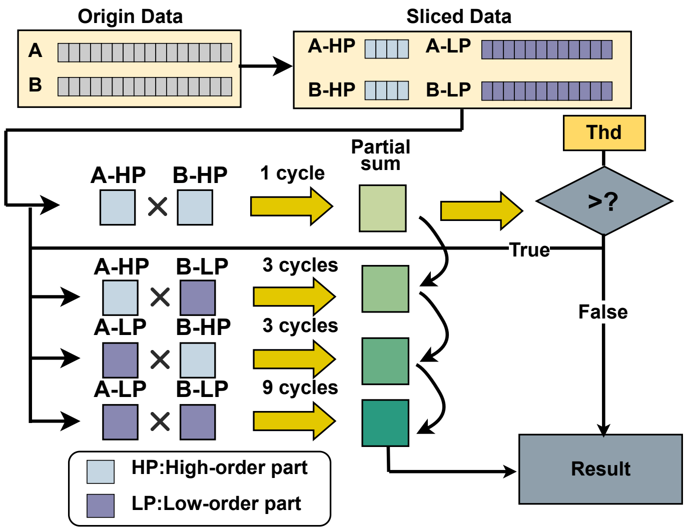

Fig. 2: An Example of OPS Computation Flow.

This approach enables efficient computational skips for less significant values, substantially reducing overall computational costs without compromising model accuracy. To optimize the trade-off between efficiency and accuracy, we determine the optimal threshold and preset values through offline layer-wise validation, minimize any potential accuracy loss while maximizing computational savings.

III. OPS Execution Flow

Figure 2 illustrates the execution flow of OPS computation, which comprises five steps:

- **Data Slicing:** Input data is partitioned into 4-bit high-order and 12-bit low-order components. This division occurs directly in storage using dedicated buffers, eliminating the need for additional encoding processes.
- **Weight Preparation:** Weights are pre-partitioned into high-order and low-order components and stored in their respective buffers. These pre-processed weights are then fixed in the systolic array prior to computation.
- **Early Bird Computation:** The partitioned inputs and weights enter the PE array in 4-bit format. For each computation set, OPS initially processes the high-order parts and determines whether to activate the early termination mechanism. If activated, the remaining computations in the set are bypassed, and predetermined values are used to populate the partial sum buffers.
- **Partial Sum Accumulation:** Rather than directly aggregating results from each case, OPS accumulates partial sums in a dedicated buffer.
- **Precision-Aware Reduction:** OPS performs a reduction operation on the accumulated partial sums to generate 16-bit output results. These results are then stored in an interleaved format, with the 4-bit high-precision and 12-bit low-precision parts separated.

IV. Evaluation

Accuracy/Perplexity. Table I shows the evaluation results of OPS. These results demonstrate that even with most computations skipped through the early bird mechanism, OPS can maintain model accuracy while providing acceleration for LLMs.

TABLE I: Accuracy and Perplexity of LLMs with PTQ. Lower perplexity (PPL) indicates better performance.

Model	BERT	GPT2-medium	GPT2-XL	OPT-6.7B	LLaMA-7B
Dataset	SST-2 (Acc.)	Wikitext-103 (PPL)			
FP32	93.23	22.76	16.94	10.86	5.68
Olive 8/16-bit	92.88	25.42	19.49	18.34	36.47
OPS 8/16-bit	92.77	23.55	18.15	13.11	8.98

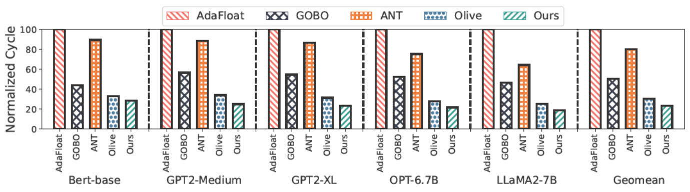

Fig. 3: Comparison of the normalized latency in different designs.

Performance. Figure 3 illustrates the normalized total execution cycles across different accelerators for five networks. OPS demonstrates superior performance improvement, achieving substantial speedups with average improvements of $1.30\times - 4.31\times$ over other baselines. Notably, OPS achieves a $3.70\times$ speedup over ANT on GPT2-XL. OPS stands out by fully exploiting inherent sparsity through precision-slicing and achieving significant computational speedup via the early-bird mechanism, without additional encoding and decoding overhead.

V. Conclusion

This paper introduces OPS, an outlier-aware precision-slice framework designed to enhance the efficiency of LLM inference. The key insight of OPS is to exploit the inherent sparsity and contribution within the computation of higher-order parts through the bit-slicing scheme. By integrating an early-bird mechanism, OPS selectively bypasses unnecessary computations for better performance. Evaluations show that OPS delivers significant performance gains with only a negligible accuracy loss.

References

[1] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.

[2] F. Liu *et al.*, "Improving neural network efficiency via post-training quantization with adaptive floating-point," in *ICCV*, 2021.

[3] F. Liu, N. Yang *et al.*, "Inspire: Accelerating deep neural networks via hardware-friendly index-pair encoding," in *DAC*, 2024, pp. 1–6.

[4] F. Liu, N. Yang, Z. Wang *et al.*, "Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding," in *HPCA*. IEEE, 2024, pp. 1029–1042.

[5] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.