## Overcoming the Efficiency Barrier in LLM Inference:

## The Role of Low-Precision Quantization Research

###### Introduction

Large language models (LLMs) have revolutionized artificial intelligence by achieving state-of-the-art performance across diverse domains, including machine translation [1], contextual reasoning [2], and creative content generation [3]. These transformer-based architectures leverage massive parameter scales—now routinely exceeding hundreds of billions—to capture intricate linguistic patterns. However, this exponential growth introduces severe deployment bottlenecks that threaten their practical viability.

Central to this challenge are the prohibitive resource requirements of LLM inference. As exemplified by models like DeepSeek-R1 (671B parameters) [4], even at FP16 precision (current standard for deployment), memory demands reach 1,342 GB. This vastly surpasses the capacity of cutting-edge hardware such as NVIDIA’s B200 GPU (192 GB) [5], creating an operational chasm between model capabilities and accessible infrastructure. The implications extend beyond high-end systems: Multi-GPU parallelism escalates costs and complexity, real-time applications (e.g., conversational agents) suffer from slow token generation, and megawatt-scale data center consumption raises sustainability concerns.

The essay is structured as follows. In section II, we examine the operational principles of LLMs and analyze inherent bottlenecks in their inference process. Section III presents current low-precision quantization techniques that accelerate LLM inference by reducing computational demands. Section IV discusses practical deployment scenarios for mainstream low-precision quantization methods. Finally, section V concludes this essay.

###### Background and Challenges

Current LLMs predominantly utilize Transformer-based architectures, as illustrated in Figure 1. Each Transformer block comprises two core components: a multi-head attention layer and a feed-forward network (FFN) layer. The attention layer incorporates four linear projection operations: query, key, value, and output projections. The FFN layer typically consists of two linear transformations, though certain LLM variants (e.g., Llama family) implement three sequential linear operations. Previous literature has shown that linear transformations in transformer blocks dominate over 70% of the overhead [6].

The inference workflow of LLMs operates in two distinct phases: the prefill phase and the decode phase. The prefill phase serves as the context initialization stage, processing the entire input prompt in parallel to establish hidden states for subsequent generation. This process is compute-bounded and generates a certain amount of KV cache as it processes the entire input sequence. The decode phase performs progressive token-by-token prediction using these initialized states, operating in strict auto-regressive fashion where each generated token becomes input for the next step [6]. In the long-context generation task, this phase will generate a large number of KV caches, which will bring extremely high challenges to

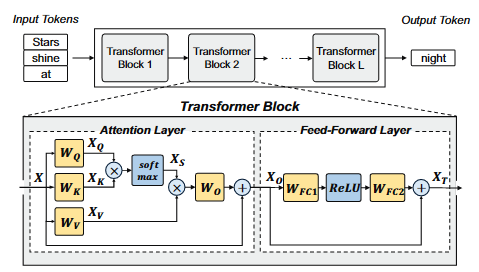


Figure 1: Illustration of the Transformer block architecture.

memory phase will generate a large number of KV caches, which will bring extremely high challenges to memory capacity and bandwidth. As a result, the decode phase encounters crippling memory-bandwidth limitations under typical low-batch scenarios. Empirical studies confirm that KV cache loading overhead for billion-parameter models consumes over 90% of latency per token due to constrained high-bandwidth memory (HBM) throughput on modern accelerators.

These dual constraints—compute-bound during prefill and memory-bound during decoding—collectively limit deployment efficiency across cloud-to-edge environments, establishing an imperative for architectural optimizations.

###### The Solution: Low-Precision Quantization

Among existing optimization techniques, quantization has emerged as one of the most effective methods for reducing the inference cost of LLMs [7, 8]. Traditionally, neural network quantization can be categorized into two approaches: Post-Training Quantization (PTQ) [7, 9-21] and Quantization-Aware Training (QAT) [7, 22-24]. While QAT generally achieves superior performance, retraining LLMs is often impractical in real-world deployment due to their massive scale and prohibitive computational overhead [11]. Consequently, PTQ is more widely adopted in practice.

Current quantization approaches for LLMs can be categorized into three types: (1) Weight-only quantization: It primarily addresses memory-bound General Matrix-Vector Multiply (GEMV) operations in the decoding phase [9, 17, 24]. (2) Weight-Activation quantization: It simultaneously addresses both memory-bounded GEMV operations in the decoding phase and compute-bound General Matrix Multiply (GEMM) pressure during the prefill phase [10, 12-15, 19-21]. (3) Key-Value (KV) cache quantization: It targets the decoding phase for long-sequence or large-batch scenarios [25, 26], where the memory overhead from KV cache can exceed that of the model weights themselves [27]. The KV cache can be conceptually regarded as a specialized form of activation that persists across sequential decoding steps. Unlike transient activations in conventional forward passes, the KV cache retains computed key-value states from previous tokens to enable efficient autoregressive generation. Among these, weight-activation quantization provides the most comprehensive benefits, as it simultaneously targets memory-bound and compute-bound bottlenecks.

###### Real-World Application

Native support for low bit-width formats is now prevalent in AI-focused hardware. For example, NVIDIA has introduced INT8 integer type in tensor core since its Turing architecture [28]. Particularly, 8-bit quantization necessitates a scale factor for each tensor and has demonstrated noteworthy impact when the number of model parameters is relatively small [29], [30]. However, when dealing with Transformer-based Large Language Models (LLMs), the model’s performance is easily affected by a very small portion (< 0.1%) of outliers [9], [18]. For example, regular quantization methods fail at a scale of 6.7B parameters for OPT model [18].

Additionally, tensor-level scaling has been shown to be insufficient for Transformer-based LLMs due to their limited dynamic range. Various methods have been proposed to address outliers, with one common approach being to handle normal values and outliers differently [11, 16, 22]. For example, OliVe [11] adopts an outlier-victim pair (OVP) quantization to sacrifice normal values to accommodate outliers locally. GOBO [16] and OLAccel [31] use coordinate list to distinguish outliers from normal values. However, such progress not only leads to unaligned memory storage and accesses but also causes large area overheads due to the extra outlier controller.

Another approach is to use new data formats to simultaneously accommodate outliers and regular values. Research has shown that using micro-scaled data formats, which associate scaling factors with fine-grained sub-blocks of a tensor, is a promising method to quantize Transformer-based LLMs [10, 12, 13, 23] and even more effective in the sub-8 bit regime [12, 13, 23]. However, these efforts are all optimized for existing GPUs, which are architecturally not compatible with these new data formats, often requiring complex dynamic quantization algorithms such as data recognition and reordering. For example, the main loop of ATOM [13] is therefore dominated by slow CUDA core operations instead of fast tensor core operations, resulting in a similar actual inference speed as FP16. Microscaling (MX) data format [32] has been proposed as the first open standard for a family of microscaled data formats aimed at deep learning inference. However, the deployment of MX still has a long way to go through the combined efforts of industries and academia.

Despite recent progress, achieving accurate and efficient low-bit quantization (e.g., 4-bit quantization) remains challenging due to the presence of activation outliers [8]. These rare but large-magnitude values disproportionately impact scaling factors, causing significant quantization errors for normal values in the same group. While group-wise quantization [9, 13, 17] and mixed-precision techniques [16, 22, 31] have been proposed to address this, they often suffer from precision-efficiency trade-offs, hardware overhead from FP16 scaling, and misalignment with DRAM memory structures.

###### Conclusion

Model quantization has demonstrated significant efficacy in reducing memory footprint and computational demands through lower-bit numerical representations, enabling more scalable and efficient LLM deployment. Nevertheless, achieving optimal Pareto efficiency between model accuracy and hardware performance remains an ongoing challenge for contemporary quantization methodologies.

###### Reference

[1] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal et al., “Language models are few-shot learners,” arXiv preprint arXiv:2005.14165, vol. 1, p. 3, 2020.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat et al., “Gpt-4 technical report,” arXiv preprint arXiv:2303.08774, 2023.

[3] Microsoft, “Github copilot,” https://github.com/features/copilot, 2023.

[4] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi et al., “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” arXiv preprint arXiv:2501.12948, 2025.

[5] N. Corporation, “Nvidia blackwell b200 gpu,” https://www.nvidia.com/ en-us/data-center/dgx-b200/, 2024.

[6] Z. Yuan, Y. Shang, Y. Zhou, Z. Dong, Z. Zhou, C. Xue, B. Wu, Z. Li, Q. Gu, Y. J. Lee et al., “Llm inference unveiled: Survey and roofline model insights,” arXiv preprint arXiv:2402.16363, 2024.

[7] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, “A white paper on neural network quantization,” arXiv preprint arXiv:2106.08295, 2021.

[8] 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.

[9] 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.

[10] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, “Olive: Accelerating large language models via hardwarefriendly outlier-victim pair quantization,” in Proceedings of the 50th Annual International Symposium on Computer Architecture. IEEE, 2023, pp. 1–15.

[11] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han, “Qserve: W4a8kv4 quantization and system co-design for efficient llm serving,” arXiv preprint arXiv:2405.04532, 2024.

[12] Y. Zhao, C.-Y. Lin, K. Zhu, Z. Ye, L. Chen, S. Zheng, L. Ceze, A. Krishnamurthy, T. Chen, and B. Kasikci, “Atom: Low-bit quantization for efficient and accurate llm serving,” Proceedings of Machine Learning and Systems, vol. 6, pp. 196–209, 2024.

[13] C. Xue, C. Zhang, X. Jiang, Z. Gao, Y. Lin, and G. Sun, “Oltron: Algorithm-hardware co-design for outlier-aware quantization of llms with inter-/intra-layer adaptation,” in Proceedings of the 61st ACM/IEEE Design Automation Conference, 2024, pp. 1–6.

[14] J. Koo, D. Park, S. Jung, and J. Kung, “Opal: Outlier-preserved microscaling quantization accelerator for generative large language models,” in Proceedings of the 61st ACM/IEEE Design Automation Conference, 2024, pp. 1–6.

[15] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos, “Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference,” in 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2020, pp. 811–824.

[16] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, “Gptq: Accurate post-training quantization for generative pre-trained transformers,” arXiv preprint arXiv:2210.17323, 2022.

[17] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, “Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale,” Advances in Neural Information Processing Systems, vol. 35, pp. 30 318–30 332, 2022.

[18] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, “Zeroquant: Efficient and affordable post-training quantization for largescale transformers,” Advances in Neural Information Processing Systems, vol. 35, pp. 27 168–27 183, 2022.

[19] S. Ashkboos, I. Markov, E. Frantar, T. Zhong, X. Wang, J. Ren, T. Hoefler, and D. Alistarh, “Quik: Towards end-to-end 4-bit inference on generative large language models,” arXiv preprint arXiv:2310.09259, 2023.

[20] J. Lee, W. Lee, and J. Sim, “Tender: Accelerating large language models via tensor decomposition and runtime requantization,” arXiv preprint arXiv:2406.12930, 2024.

[21] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, “Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization,” in 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2022, pp. 14141433.

[22] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” Advances in Neural Information Processing Systems, vol. 36, 2024.

[23] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, “Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 12, 2024, pp. 13 355–13 364.

[24] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, S. Shao, K. Keutzer, and A. Gholami, “Kvquant: Towards 10 million context length llm inference with kv cache quantization,” Advances in Neural Information Processing Systems, vol. 37, pp. 1270–1303, 2024.

[25] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu, “Kivi: A tuning-free asymmetric 2bit quantization for kv cache,” arXiv preprint arXiv:2402.02750, 2024.

[26] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with pagedattention,” in Proceedings of the 29th Symposium on Operating Systems Principles, 2023, pp. 611–626.

[27] J. Burgess, “Rtx on—the nvidia turing gpu,” IEEE Micro, vol. 40, no. 2, pp. 36–44, 2020.

[28] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, “I-bert: Integer-only bert quantization,” in International conference on machine learning. PMLR, 2021, pp. 5506–5518.

[29] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “Q-bert: Hessian based ultra low precision quantization of bert,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, 2020, pp. 8815–8821.

[30] E. Park, D. Kim, and S. Yoo, “Energy-efficient neural network accelerator based on outlier-aware low-precision computation,” in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2018, pp. 688–698.

[31] B. Darvish Rouhani, R. Zhao, V. Elango, R. Shafipour, M. Hall, M. Mesmakhosroshahi, A. More, L. Melnick, M. Golub, G. Varatkar et al., “With shared microexponents, a little shifting goes a long way,” in Proceedings of the 50th Annual International Symposium on Computer Architecture, 2023, pp. 1–13.

[32] S. Li, X. Ning, L. Wang, T. Liu, X. Shi, S. Yan, G. Dai, H. Yang, and Y. Wang, “Evaluating quantized large language models,” arXiv preprint arXiv:2402.18158, 2024.

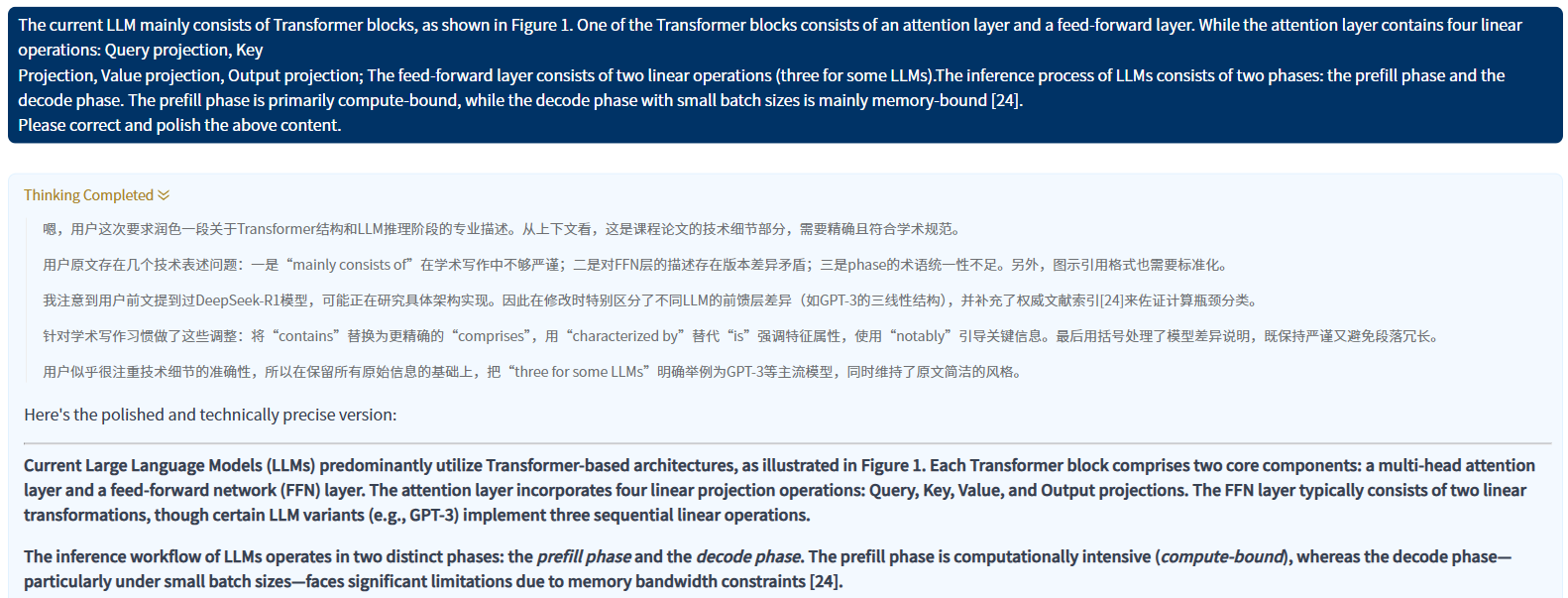
###### AI Statement

**Polish the structure (Section I):**





**Correcting errors and polishing principles section (Section II):**



**Improve the introduction logic and add two stages of action description (Section II):**



**Polish the conclusion(Section V):**

