

A Scheduling Framework for Efficient MoE Inference on Edge GPU-NDP Systems

Qi Wu¹, Chao Fang^{1,†}, Jiayuan Chen², Ye Lin¹, Yueqi Zhang¹, Yichuan Bai¹, Yuan Du¹, Li Du¹

¹School of Electronic Science and Engineering, Nanjing University, China ²China Mobile Research Institute, China
Email: {qiwu, fantasysee, yelin, zhangyueqi, yicbai}@smail.nju.edu.cn, chenjiayuan@chinamobile.com, {yuandu, ldu}@nju.edu.cn

Abstract—Mixture-of-Experts (MoE) models facilitate edge deployment by decoupling model capacity from active computation, yet their large memory footprint drives the need for GPU systems with near-data processing (NDP) capabilities that offload experts to dedicated processing units. However, deploying MoE models on such edge-based GPU-NDP systems faces three critical challenges: 1) severe load imbalance across NDP units due to non-uniform expert selection and expert parallelism, 2) insufficient GPU utilization during expert computation within NDP units, and 3) extensive data pre-profiling necessitated by unpredictable expert activation patterns for pre-fetching. To address these challenges, this paper proposes an efficient inference framework featuring three key optimizations. First, the underexplored tensor parallelism in MoE inference is exploited to partition and compute large expert parameters across multiple NDP units simultaneously towards edge low-batch scenarios. Second, a load-balancing-aware scheduling algorithm distributes expert computations across NDP units and GPU to maximize resource utilization. Third, a dataset-free pre-fetching strategy proactively loads frequently accessed experts to minimize activation delays. Experimental results show that our framework enables GPU-NDP systems to achieve 2.41× on average and up to 2.56× speedup in end-to-end latency compared to state-of-the-art approaches, significantly enhancing MoE inference efficiency in resource-constrained environments.

I. INTRODUCTION

For transformer-based [1] large language models (LLMs), the Mixture-of-Experts (MoE) model has recently emerged as an efficient architectural paradigm that enables massive model scaling through sparse computation. As shown in Fig. 1, the core mechanism of MoEs involves sparse activation in the feed-forward network (FFN) stage, where only a selected subset of parameters, termed “experts,” is engaged for each token [2], [3], differentiating it from dense models [4]–[7] that activate all parameters. However, this efficiency comes at substantial memory costs, as expert parameters typically ranging from tens to hundreds of billions [8], far exceeding consumer-grade GPU capacity. For instance, RTX 5080, a modern high-end consumer-grade GPU, offers 16GB VRAM [9], which is insufficient to host many large MoE models, e.g., Qwen3-30B-A3B [10] whose parameters can occupy more than 60GB. This memory capacity gap in Fig. 1, presents a critical challenge for edge LLM deployment with limited resource budget.

[†]Corresponding author. This work was funded in part by the National Key Research and Development Program of China under Grant 2022YFB4400900, in part by the Strategic Industries and Key Technologies Project of Jiangsu Province under Grant BE2023020-3, in part by the Basic Research Program of Jiangsu Province under Grant BK20243042, and in part by the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute.

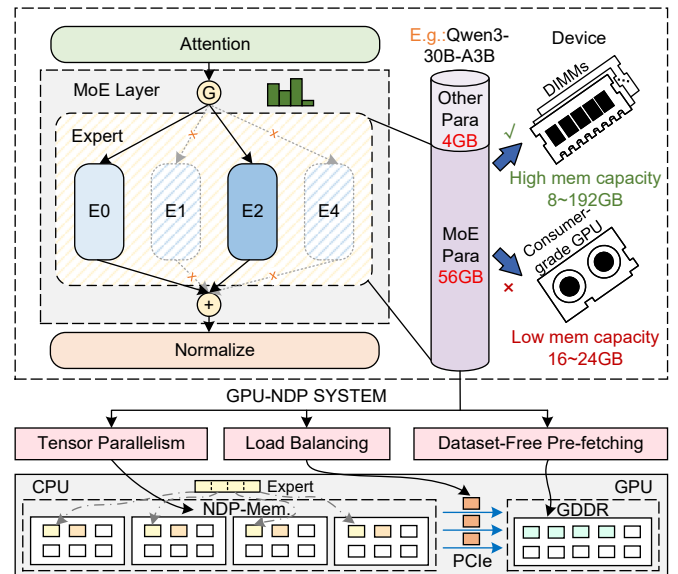


Fig. 1. The rising memory gap between consumer-grade GPU VRAM and large MoE models motivates GPU-NDP systems as a promising cost-effective edge solution. Our scheduling framework enables efficient MoE inference on such systems via tensor parallelism, load balancing, and dataset-free pre-fetching.

To address this challenge of memory capacity, current research follows mainly three different technical paths. Two of them are CPU-based solutions. The first one leverages host system memory, i.e., CPU DRAM as an extended GPU memory pool, offloading inactive expert parameters and transferring them back when needed [11]–[13]. The second one incorporates heterogeneous computing, where both parameters and computations are partially delegated to the CPU to reduce transmission overhead [14]–[16]. Though these approaches could mitigate transmission latency through static pre-fetching [12] or multi-batch strategies [15], they remain fundamentally constrained by memory bandwidth and computational capability of CPU. Moreover, static pre-fetching requires extensive pre-runs with calibration data to generate expert usage patterns, limiting its adaptability to edge dynamic scenarios.

The third category explores near-data processing (NDP) systems by situating computation proximate to data storage. [17], [18]. These systems can address memory capacity constraints and substantially reduce data transfer latency over the PCIe bus by integrating computation capabilities through pluggable dual in-line memory modules (DIMMs) or modified graphics double data rate (GDDR) into GPU memory systems. However, cur-

rent GDDR-based approaches incur substantial implementation costs [18], making them less suitable for cost-effective edge deployments. More critically, existing scheduling strategies for GPU-NDP systems suffer from several limitations. First, expert parallelism implementations across multi-node systems [17], [18] lead to inefficient expert distribution during single-batch inference. Furthermore, heterogeneous scheduling schemes like Duplex [18] fail to achieve balanced computation between GPU and NDP devices due to consistent arithmetic intensity across experts during the decode stage, while MoNDE [17], despite introducing load-balancing strategies, relies on periodic adjustments based on historical profiles, resulting in sub-optimal efficiency for dynamic edge workloads. Additionally, the reliance on calibration-based expert prefetching strategies [17] becomes inaccurate in edge inference scenarios where input distributions differ from calibration datasets, leading to reduced efficiency.

To address the above limitations, this paper proposes a novel scheduling framework for accelerating MoE inference on edge GPU-NDP systems, specifically targeting localized environments such as personal workstations, as shown in Fig. 1. The framework is designed for edge deployments with NDP-enabled DIMM configurations, which are preferred over HBM-PIM due to their superior cost-effectiveness and reduced integration complexity. And tensor parallelism is employed to partition expert parameters across devices. Unlike existing approaches that rely on dataset-specific expert activation heatmaps, we dynamically observe expert activation patterns during initial prompt processing phase, i.e., *prefill*, and proactively pre-fetch experts predicted to be active in the subsequent token generation phase, i.e., *decode*, to the GPU. Additionally, when no activated expert resides on the GPU, our load-balancing mechanism transmits only partial expert parameters back to GPU, enabling simultaneous utilization of both GPU and NDP-DIMM resources for computation. Our key contributions are summarized as follows.

- 1) A scheduling framework is proposed for efficient MoE inference on GPU-NDP systems that integrates a consumer-grade GPU and cost-effective NDP-DIMMs. It is specifically optimized for single-batch inference scenarios common in edge deployment.
- 2) A dynamic hybrid scheduling strategy is proposed to enhance the efficiency of MoE inference, uniquely combining tensor parallelism with load balancing mechanisms and dataset-free expert pre-fetching.
- 3) Comprehensive evaluation across four popular MoE models is conducted, showing up to 2.56 \times speedup in end-to-end latency compared with the state-of-the-art (SOTA) MoNDE [17] and demonstrating the effectiveness of our approach for edge MoE inference.

II. BACKGROUND AND RELATED WORK

A. Demand for Edge MoE Deployment

Local deployment of MoE models is driven by requirements for offline availability, strengthened data privacy, and user-specific personalization without cloud dependency [19]–[22]. However, the substantial parameter footprint of modern MoE

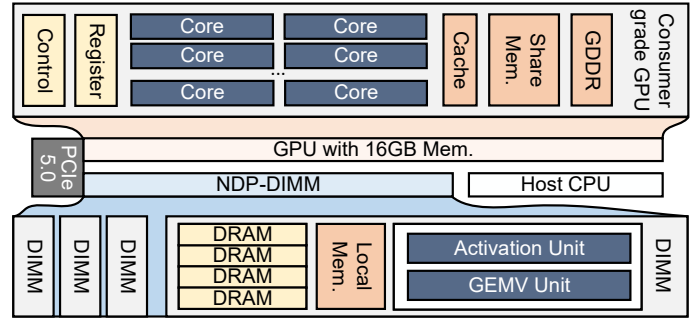


Fig. 2. Overview of GPU-NDP DIMM System.

models, such as Qwen3-30B-A3B [10] which exceeds 60GB, significantly impedes edge deployment under tight memory and I/O budgets.

Empirical evidence reveals that expert activation follows highly uneven patterns: only a small fraction of experts are frequently activated while the majority remain idle for most tokens. Prior systems [12], [15] exploit this sparsity through expert pre-fetching and inter-batch weight reuse to amortize loading overheads across large batches for efficient inference. However, edge settings typically involve single-user interaction with a batch size of 1 [23], eliminating inter-batch amortization opportunities. Moreover, traditional pre-fetching [12] relies on offline profiling with calibration datasets to determine expert usage patterns, leading to suboptimal efficiency in dynamic edge scenarios. This mismatch between calibration data and actual workloads often results in unused expert transfers during single-batch inference. Consequently, weight traffic between processors and memory becomes the dominant bottleneck in low-batch edge inference. This motivates exploring NDP approaches that bring computation closer to where expert weights reside, rather than continuously transferring large weight matrices.

B. Near-Data Processing for MoE Model Acceleration

NDP methods [17], [18], [24]–[36] address the data movement bottleneck by placing computation close to or within memory arrays. In MoE inference, this approach is particularly effective since traffic is dominated by expert weights rather than activations. For example, in single-batch scenarios, the expert weights activated per layer are 672 MB in Mixtral-8x7B [37], while the activations are only 0.0078 MB. By executing expert computation where weights reside, NDP allows only lightweight activations to be transported.

Currently, NDP platforms span HBM-PIM solutions and commodity-module designs based on DIMM or LPDDR devices. HBM-PIM methods show strong performance on suitable workloads [26], [27], but its reliance on specialized, costly stacked memory limits adoption in cost-sensitive, locally deployed systems. Instead, DIMM/LPDDR-based NDP leverages standard modules and offers a more economical, scalable path for edge deployments [17], [30]. However, existing commodity solutions have significant limitations for edge MoE inference. LP-Spec [30] employs LPDDR-NDP with NPU to accelerate dense models via dynamic scheduling that coordinates the computation of the attention and FFN. However, its scheduling

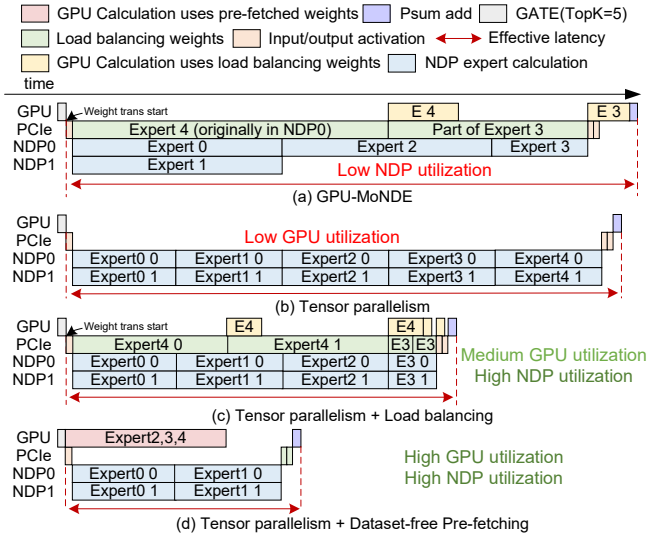


Fig. 3. Comparison between MoE workflow scheduling in the decode stage. The proposed scheduling framework is based on (b) tensor parallelism and supplemented by (c) load balancing and (d) dataset-free pre-fetching.

is triggered only during token pruning and does not encompass MoE models. By contrast, MoE models are explicitly targeted in MoNDE [17], where expert computations are offloaded to LPDDR-NDP units while retaining other computations on a GPU. It primarily considers a single large-memory NDP device and scales poorly to multi-NDP configurations under single batch, where inter-device workload imbalance significantly lower utilization. These limitations highlight the need for more effective scheduling strategies tailored to multi-NDP edge deployment scenarios, which motivates our proposed framework.

III. THE PROPOSED FRAMEWORK

A. Framework overview

Fig. 2 presents the overview of the proposed system framework, which is based on a single consumer-grade GPU and multiple NDP-DIMMs. Our framework adopts the central buffer-based NDP-DIMM architecture [18], [24], [28], [38], where an integrated NDP core on each DIMM processes locally stored data. Under this GPU-NDP DIMM system, the computational workload is distributed across different processing units. The dominant MoE computations in FFN layers are delegated to DIMM-based NDP units, while all remaining components, including the attention mechanism, are processed on the GPU. The host CPU is reserved for task scheduling and coordination. While our analysis is grounded in this specific hardware configuration, the framework is broadly applicable to the other GPU-NDP systems. To optimize MoE inference, we propose three collaborative innovations: (1) a tensor parallelization strategy that distributes expert weights efficiently across NDP units; (2) a load balancing scheduling strategy that adapts to token routing patterns; (3) a dataset-free pre-fetching strategy that eliminates prior data profiling. Fig. 3 shows these strategies and their incremental benefits over the SOTA MoNDE [17].

B. Tensor parallelism

The conventional expert parallelism used by MoNDE [17] allocates entire experts to distinct NDPs. However, the dynamic

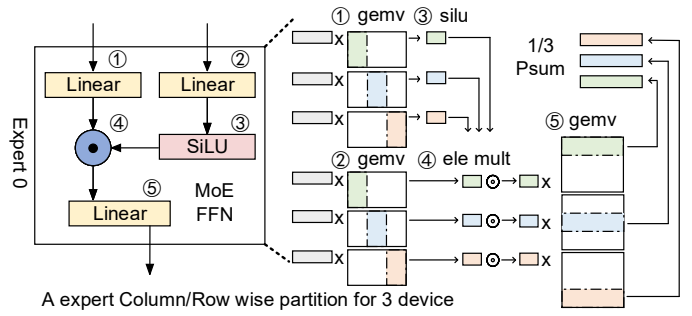


Fig. 4. The computing process at the MoE layer with our introduced tensor parallelism. The left side is the common structure of the MoE Expert FFN, and the right side is the deployment of one expert to three computing devices.

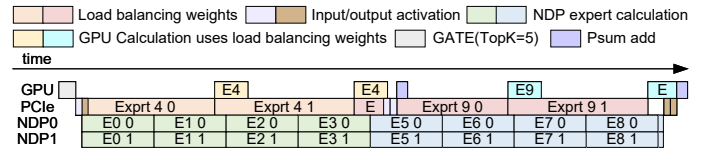


Fig. 5. MoE workflow scheduling in the prefill stage within our framework.

routing mechanism inherent in MoE models does not guarantee uniform expert activation across NDPs, resulting in suboptimal resource utilization as demonstrated in Fig. 3a. In this case, four activated experts are deployed in NDP0 whereas only one resides in NDP1, rendering NDP0 the critical path. Although partial workloads have been offloaded to the GPU, the critical path remains lengthy.

To mitigate this inefficiency, we propose implementing tensor parallelism across NDPs during the MoE computation stage. This strategy has historically been overlooked in the MoE scheduling research, largely due to the perceived high communication overhead of tensor parallelism. In single-batch, multi-NDP settings, however, the load imbalance introduced by expert parallelism outweighs the communication cost of tensor parallelism. As depicted in Fig. 3b, tensor parallelism uniformly partitions each expert's computation across all available NDPs. Consequently, regardless of which expert is activated, each NDP is assigned a portion of the computation. Fig. 4 illustrates our tensor parallelism strategy employed in MoE layer computations, using three NDP devices as an example. We utilize a two-stage partitioning approach. The GEMV operations in the first two linear layers are partitioned by columns, while that in the last linear layers adopt row-wise partitioning. This approach efficiently distributes computational workload and memory requirements across devices to generate the final partial sum. However, while tensor parallelism achieves high utilization of NDPs in Fig. 3b, the GPU remains idle during the decode stage, leaving room for further optimization. This observation motivates the need for load balancing between GPU and NDP to fully leverage both computational resources.

C. Load Balancing

To utilize idle GPU resources during NDP computation, we propose the load balancing strategy shown in Fig. 3c and Fig. 5. Existing strategies like MoNDE [17] suffer from two limitations: they ignore the imbalanced workload distribution

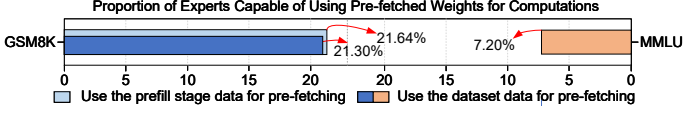


Fig. 6. Utilization of prefetched weights in Qwen3-30B-A3B [10] is compared between analyses derived from the full calibration datasets (GSM8K, MMLU) and from dataset-free prefill-stage data. The comparison is conducted using 20 randomly sampled GSM8K inputs.

among activated experts and oversimplify balance conditions by only equating NDP computation time with weight transmission time, neglecting differences in the number of input tokens between the prefill and decode stages. Furthermore, they use a scaling factor based on historical data to correct the load balancing equation.

We reformulate the balance condition based on tensor parallelism. The key insight is that the E_g value, denoting the number of experts assigned to the GPU, should be determined to balance the total time spent on weight transmission and GPU computation against the total time for NDP calculation and input/output activation data transmission, rather than requiring adjustment via a periodic scaling factor as in MoNDE [17]. We also define T_g as GPU expert computation latency, T_n as NDP-DIMM expert computation latency after tensor parallelism, T_w as expert transmission time, T_a as activation transmission time. N represents the number of enabled NDP-DIMMs, and S represents the sequence length. Since the last transmission cannot be hidden, $S - 1$ is actually used. The workload parameters include TopK as the number of activated experts per layer, E_n as the number of experts on NDP-DIMM, where $E_n + E_g = \text{TopK}$. Note that most computations on the GPU are overlapped by weight transfer; therefore, $E_{g'}$ denotes the unhidden computing time. For analytical tractability, $E_{g'}$ is defined as one N -th of the fractional part of E_g , reflecting the fraction of GPU expert computation that is not overlapped by N -way weight transfer; when E_g is an integer multiple of N , a nominal value of $1/N$ is used instead.

Load balancing condition in decode stage:

$$T_w \cdot E_g + T_g \cdot E_{g'} = (N + 1) \cdot T_a + T_n \cdot E_n. \quad (1)$$

Load balancing condition in prefill stage:

$$T_w \cdot E_g + T_g \cdot E_{g'} + (S - 1) \cdot T_a \cdot N = (N + 1) \cdot T_a + T_n \cdot E_n. \quad (2)$$

The proposed load balancing conditions are formulated differently for decode and prefill stages due to their distinct computational characteristics. Equation (1) defines the balance condition for the decode stage, while Equation (2) accounts for sequence length dependencies in the prefill stage. Using these equations, the balanced expert E_g can be calculated. As illustrated in Fig. 3c, load balancing enables a further improvement in the acceleration of the MoE stage. While this approach optimizes expert allocation, the frequent weight transfers required by dynamic load balancing may be limited by PCIe bandwidth, prompting us to further optimize runtime transmission overhead.

D. Dataset-free Pre-fetching

Pre-fetching expert weights into GPU memory before they are needed can significantly reduce PCIe transfer latency and

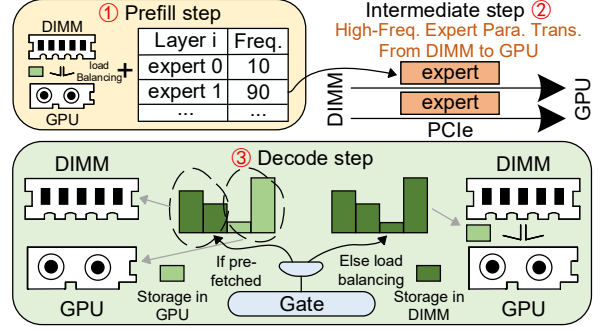


Fig. 7. Dataset-free pre-fetching strategy. It is divided into three steps: ①Prefill step, ②Intermediate step, and ③Decode step.

improve GPU utilization. Conventional pre-fetching strategies [12], however, typically rely on a pre-profiling of a calibration dataset to ascertain the activation patterns of experts, incurring hundreds to thousands of extra inference passes and leading to dataset-dependent variability in the utilization of pre-fetched weights, as shown in Fig. 6. Although [14] mentions that active experts have similarities in the prefill and decode stages, it still used the calibration dataset for initialization and introduced accuracy errors. Our strategy eliminates this limitation through a three-step pipeline as shown in Fig. 7, which dynamically learns expert activation patterns during actual inference execution without calibration data. It is divided into three steps: ①prefill, ②intermediate, and ③decode. The core innovation, as illustrated in Fig. 6, leverages activation patterns observed in the prefill stage to guide pre-fetching decisions for the decode stage, transforming costly offline profiling into lightweight runtime learning that adapts to actual workload characteristics.

①**Prefill step: dynamic information collection.** As shown in Fig. 7①, during the prefill step, the system processes the input prompt while simultaneously collecting expert activation statistics in real-time. As FFN computations execute using the load balancing strategy from Equation (2), the system continuously monitors which expert modules are activated and tracks their frequency patterns. This dynamically collected activation data forms an instantaneous, task-specific expert table without requiring offline calibration datasets, which serves as a direct basis for pre-fetching decisions in the subsequent step. Its pre-fetched expert utilization rate is even higher than that achieved with calibration datasets, as shown in Fig. 6.

②**Intermediate step: weight pre-fetching.** Based on the activation patterns identified during the prefill step, this intermediate step selectively pre-fetches the most frequently activated experts into GPU memory, as depicted in Fig. 7②. The number of experts to be pre-fetched is determined by available GPU memory to ensure optimal resource utilization: at each layer, x experts are pre-fetched, with x initialized at 1 and incremented until no additional experts can be accommodated by the GPU. Since a single, small data transfer relative to the overall inference workload is performed during expert pre-fetching, minimal overhead is incurred. In parallel, the remaining critical experts are staged for immediate access during decode step.

③**Decode step: adaptive computation execution.** As il-

TABLE I
GPU-NDP SYSTEM CONFIGURATION

GPU configuration		NDP-DIMM configuration	
Frequency	2.30 GHz	DDR4 type	3200 MT/s, 32GB
GDDR	16GB GDDR7	Ranks	4 / DIMM
SM Count	84	Bankgroups	8 / Rank
Tensor Cores	336	Banks	4 / Bankgroup
Interface	PCIe 5.0	Multipliers	64 / DIMM

TABLE II
WORKLOAD CHARACTERISTICS OF MoE-BASED MODELS

Model	E Para.	#E	TopK	Hidden	Interm.	#Layer
DeepSeek-MoE [41]	15.4B	64	6/2 ^Δ	2048	1408	27/28*
Qwen3-30B-A3B [10]	29.0B	128	8	2048	768	48
Phi-3.5-MoE [42]	40.3B	16	2	6400	4096	32
Mixtral-8x7B [37]	42.0B	8	2	4096	14336	32

^Δ6/2 indicates the activation of 6 routing experts and 2 shared experts.
*27/28 indicates that 27 out of 28 layers are MoE layers.

illustrated in Fig.7^③, this step operates as follows: if the activated experts have been pre-fetched to the GPU, computation proceeds on the GPU; otherwise, non-prefetched experts are executed within the NDPs, as shown in Fig.3d. However, maximizing the number of experts computing on the GPU does not necessarily yield optimal performance. A balance constraint must be maintained where GPU compute time should equal the combined NDP compute and transfer times. According to Equation (3), E_{max} can be estimated, which is the maximum number of experts that can be computed on the GPU. In addition, when none of the activated experts are pre-fetched, the framework seamlessly switches to the standard load balancing strategy to handle the computing requests, as presented in Fig. 3c. This fault-tolerant backup design ensures the maximum utilization of GPUs and NDP-DIMMs.

Maximum schedulable expert parameters to GPU:

$$T_g \cdot E_{max} = (\text{TopK} - E_{max}) \cdot T_n + (1 + N) \cdot T_a. \quad (3)$$

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

1) **System:** To evaluate our proposed scheduling framework, we employ a GPU-NDP system architecture as shown in Fig. 2. The system comprises a consumer-grade NVIDIA RTX 5080 GPU [9] with 16GB GDDR memory and PCIe 5.0 x16, paired with an Intel i7-14700 processor [39] that provides up to 192GB of system memory with a maximum bandwidth of 89.6 GB/s. The GPU-NDP system supports up to six 32GB NDP-DIMMs, each offering an internal bandwidth of 102.4GB/s as detailed in Table I. For performance evaluation, we utilize AttAcc! [26] to simulate GPU performance characteristics and employ a modified version of Ramulator 2.0 [40] to evaluate the performance of the DIMM devices with computing capabilities, where the simulation tool’s accuracy has been verified for GPU and PIM operations in [40].

2) **Models:** We use 4 representative MoE models from the HuggingFace repository for evaluation: DeepSeek-MoE [41], Qwen3-30B-A3B [10], Phi-3.5-MoE [42], Mixtral-8x7B [37]. These models exhibit diverse architectural characteristics, as

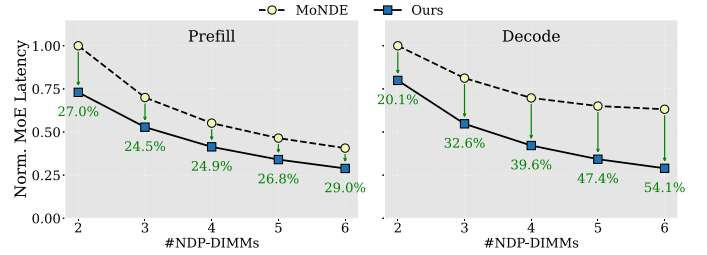


Fig. 8. The normalized MoE latency of the prefill and the decode stage in comparison with SOTA MoNDE [17].

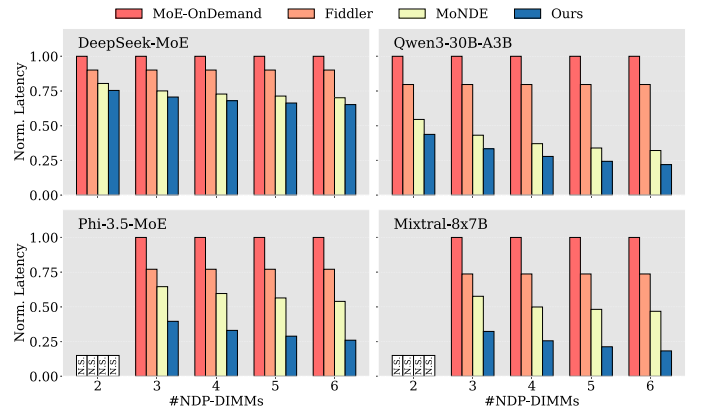


Fig. 9. End-to-end model performance with varying number of NDP-DIMMs normalized to MoE-OnDemand [11]. N.S. indicates “Not supported”, meaning that this configuration is unable to accommodate all the MoE parameters.

summarized in Table II. The number of activated experts per layer ranges from 2 to 8, and the total number of experts per layer varies from 8 to 128. Notably, DeepSeek-MoE [41] incorporates shared experts that are utilized by all tokens, distinguishing it from the other three models. Given our focus on local deployment scenarios for personal devices, all experiments are conducted with a batch size of 1. The input and output sequence lengths are uniformly set to 512 tokens across all experiments to ensure consistent evaluation conditions.

3) **Baseline:** To provide comprehensive performance comparisons, we evaluate against 3 types of baseline approaches. 1) MoE-OnDemand [11]: A GPU on-demand expert weight transmission system where MoE parameters are offloaded to system memory and activated experts are dynamically loaded to the GPU during runtime. 2) Fiddler [16]: A SOTA GPU-CPU heterogeneous system that offloads MoE parameters to memory but performs expert computations on the CPU rather than the GPU. 3) MoNDE [17]: A SOTA NDP system that stores and computes MoE components using dedicated processing units, transmitting activation values rather than model weights to reduce data movement overhead. The deployment context aligns with the edge-centric scenario examined in this work.

B. Evaluation of Isolated Prefill and Decode Performance

We analyze the MoE prefill and decode performance by comparing our framework against MoNDE [17]. Fig. 8 presents the expert computing latency between MoNDE and our approach using Qwen3-30B-A3B as an example. The results demonstrate significant performance improvements, achieving an average speedup of $1.36\times$ in the prefill stage and $1.69\times$ in the decode

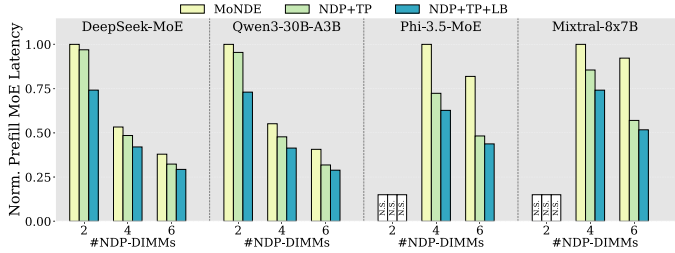


Fig. 10. The impact of tensor parallelism and load balancing on MoE computation in the prefill stage. N.S. indicates "Not supported", meaning that it is unable to accommodate all the MoE parameters.

stage, respectively. This performance gain stems from our more efficient hybrid scheduling strategy. While MoNDE distributes expert workloads cyclically based on arithmetic intensity, it fails to achieve optimal utilization across all NDP units. In contrast, our hybrid method achieves more balanced resource utilization across different scenarios. Examining the two stages in detail, the decode stage shows particularly pronounced gains compared with the prefill stage. In the prefill stage, although concurrent multi-token processing under the MoNDE scheme yields a relatively uniform distribution of activated experts across NDP units, greater speedup is nevertheless achieved by our framework through even more balanced participation of NDPs. In the decode stage, where a single token is generated serially, significant load imbalance between NDPs is observed in the MoNDE scheme. In contrast, our approach maintains the same load across all NDPs, leading to higher efficiency.

C. Evaluation of End-to-End Performance

We then evaluate the end-to-end performance of the proposed framework with varying numbers of NDP-DIMMs. Fig. 9 presents the end-to-end latency results for four MoE models, normalized against MoE-OnDemand performance. Our framework demonstrates significant performance improvements across all baseline methods. Specifically, we achieve a maximum speedup of $5.49\times$ and $4.05\times$ compared to the parameter offloading approaches MoE-OnDemand [11] and Fiddler [16], respectively, and $2.56\times$ over the NDP baseline MoNDE [17]. Since our acceleration mainly targets MoE computations, the larger the number of parameters in the MoE layer, the more significant the acceleration our framework achieves. The effectiveness of our approach is attributed to the integration of tensor parallelism, load balancing, and pre-fetching techniques, which collectively enable efficient expert distribution across NDP-DIMMs while optimizing both NDP utilization and GPU resource allocation.

D. Ablation Study for Our Proposed Techniques

An ablation study is conducted to quantify the individual and joint contributions of our three key techniques: tensor parallelism, load balancing, and pre-fetching. We compare different combinations of these techniques against the NDP baseline, i.e. MoNDE [17]: tensor parallelism alone (NDP+TP), tensor parallelism with load balancing (NDP+TP+LB), tensor parallelism with pre-fetching (NDP+TP+PRE), and the full

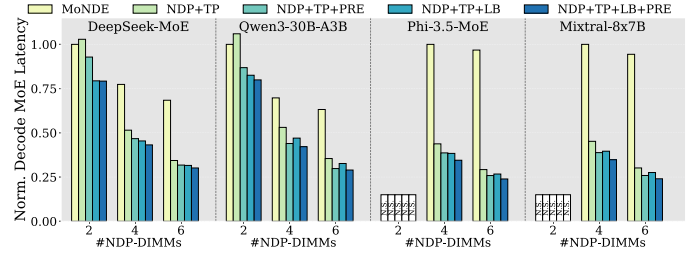


Fig. 11. The impact of tensor parallelism, load balancing, and dataset-free pre-fetching on MoE computation in the decode stage. N.S. indicates "Not supported", meaning that it is unable to accommodate all the MoE parameters.

combination (NDP+TP+LB+PRE). For all ablation configurations, we pre-fetch the most probable TopK experts per layer to GPU memory after the prefill stage for each evaluated model. However, due to GPU memory constraints, Mixtral-8x7B [37] is limited to pre-fetching only one expert per layer.

As presented in Fig. 10, during the prefill MoE stage, the NDP+TP and NDP+TP+LB strategies achieve average speedups of $1.26\times$ and $1.46\times$ over MoNDE, respectively. Since the pre-fetched experts are only applicable during the decode MoE stage, the PRE-related strategies are not shown in Fig. 10. In the decode MoE stage, as depicted in Fig. 11, an average speedup of $1.99\times$ is achieved with NDP+TP alone, while average speedups of $2.19\times$ and $2.23\times$ are achieved with NDP+TP+LB and NDP+TP+PRE, respectively, when compared with MoNDE. The combination of all three strategies (NDP+TP+LB+PRE) consistently yields the highest average speedup of $2.41\times$ across all evaluated models.

As shown in Fig. 11, two notable observations emerge from the decode MoE stage results. First, when using 2 NDP-DIMMs, the NDP+TP method performs slightly worse than MoNDE for DeepSeek-MoE and Qwen3-30B-A3B models. Individual NDP units have much lower compute capability than a GPU, making tensor parallelism less effective when NDP resources are limited, while MoNDE can leverage both GPU and NDP coordination. Second, the benefit of hybrid strategies over tensor parallelism alone decreases as we add more NDP-DIMMs. With more NDP-DIMMs, expert computation becomes much faster, shortening the time window for parameter transfer and GPU computation, which reduces the relative gains from load balancing and pre-fetching optimizations.

V. CONCLUSION

This paper presents a scheduling framework that leverages GPU-NDP systems to accelerate MoE model inference in edge deployment scenarios. The proposed framework exploits tensor parallelism to partition expert parameters across multiple NDP units, implements a load-balancing-aware scheduling algorithm to optimize resource utilization across both NDP units and GPU, and employs a dataset-free pre-fetching strategy to proactively load frequently accessed experts. Experimental results illustrate that with these collaborative innovations, our framework achieves $2.41\times$ on average and up to $2.56\times$ reduction in end-to-end latency compared to the state-of-the-art NDP-based MoNDE approach, significantly enhancing MoE inference efficiency in resource-constrained edge environments.

REFERENCES

- [1] A. Vaswani *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [2] R. A. Jacobs *et al.*, “Adaptive mixtures of local experts,” *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.
- [3] N. Shazeer *et al.*, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [4] J. Achiam *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [5] H. Touvron *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [6] S. Zhang *et al.*, “Opt: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*, 2022.
- [7] C. Fang *et al.*, “Anda: Unlocking efficient llm inference with a variable-length grouped activation data format,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1467–1481.
- [8] W. Cai *et al.*, “A survey on mixture of experts in large language models,” *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2025.
- [9] NVIDIA Corporation. (2025) GeForce RTX 5080. [Online]. Available: <https://www.nvidia.com/zh-cn/geforce/graphics-cards/50-series/rtx-5080/>
- [10] A. Yang *et al.*, “Qwen3 technical report,” *arXiv preprint arXiv:2505.09388*, 2025.
- [11] R. Y. Aminabadi *et al.*, “DeepSpeed-Inference: enabling efficient inference of transformer models at unprecedented scale,” in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2022, pp. 1–15.
- [12] Z. Fang *et al.*, “Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Volume 2, 2025, pp. 574–588.
- [13] A. Eliseev *et al.*, “Fast inference of mixture-of-experts language models with offloading,” *arXiv preprint arXiv:2312.17238*, 2023.
- [14] Y. Zhang *et al.*, “DAOP: Data-aware offloading and predictive pre-calculation for efficient moe inference,” in *Design, Automation & Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [15] S. Cao *et al.*, “Moe-lightning: High-throughput moe inference on memory-constrained gpus,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Volume 1, 2025, pp. 715–730.
- [16] K. Kamahori *et al.*, “Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models,” in *International Conference on Learning Representation (ICLR)*, 2025.
- [17] T. Kim *et al.*, “Monde: Mixture of near-data experts for large-scale sparse models,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 1–6.
- [18] S. Yun *et al.*, “Duplex: A device for large language models with mixture of experts, grouped query attention, and continuous batching,” in *57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024, pp. 1429–1443.
- [19] Y. Ji *et al.*, “Co-designing binarized transformer and hardware accelerator for efficient end-to-end edge deployment,” in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024, pp. 1–9.
- [20] S. Hadish *et al.*, “Language models at the edge: A survey on techniques, challenges, and applications,” in *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*. IEEE, 2024, pp. 262–271.
- [21] L. Huang *et al.*, “A precision-scalable risc-v dnn processor with on-device learning capability at the extreme edge,” in *29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 927–932.
- [22] F. Liu *et al.*, “Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding,” in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 1029–1042.
- [23] T. Cai *et al.*, “Medusa: Simple llm inference acceleration framework with multiple decoding heads,” in *International Conference on Machine Learning (ICML)*, 2024, pp. 5209–5235.
- [24] L. Ke *et al.*, “Recnmp: Accelerating personalized recommendation with near-memory processing,” in *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 790–803.
- [25] Y. Kwon *et al.*, “Tensordimm: A practical near-memory processing architecture for embeddings and tensor operations in deep learning,” in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, pp. 740–753.
- [26] J. Park *et al.*, “Attacc! unleashing the power of pim for batched transformer-based generative model inference,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Volume 2, 2024, pp. 103–119.
- [27] L. Wu *et al.*, “PIMoE: Towards efficient moe transformer deployment on npu-pim system through throttle-aware task offloading,” in *Proceedings of the 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025.
- [28] L. Liu *et al.*, “Make llm inference affordable to everyone: Augmenting gpu memory with ndp-dimm,” in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1751–1765.
- [29] F. Devaux, “The true processing in memory accelerator,” in *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 2019, pp. 1–24.
- [30] S. He *et al.*, “LP-Spec: Leveraging lpdrr pim for efficient llm mobile speculative inference with architecture-dataflow co-optimization,” in *Proceedings of the 44rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.
- [31] Q. Jiang *et al.*, “Ndpage: Efficient address translation for near-data processing architectures via tailored page table,” in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [32] S. Liang *et al.*, “Hyqa: Hybrid near-data processing platform for embedding based question answering system,” in *2024 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2024, pp. 1–6.
- [33] C. Zhang *et al.*, “Near-memory parallel indexing and coalescing: Enabling highly efficient indirect access for spmv,” in *2024 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2024, pp. 1–6.
- [34] Y. Zhao *et al.*, “Um-pim: Dram-based pim with uniform & shared memory space,” in *ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 644–659.
- [35] Y. Chen *et al.*, “Bramac: Compute-in-bram architectures for multiply-accumulate on fpgas,” in *IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2023, pp. 52–62.
- [36] D. Kim *et al.*, “An overview of processing-in-memory circuits for artificial intelligence and machine learning,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 12, no. 2, pp. 338–353, 2022.
- [37] A. Q. Jiang *et al.*, “Mixtral of experts,” *arXiv preprint arXiv:2401.04088*, 2024.
- [38] J. Cong *et al.*, “Aim: accelerating computational genomics through scalable and noninvasive accelerator-interposed memory,” in *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, 2017, pp. 3–14.
- [39] Intel Corporation. (2025) intel-core-i7-14700. [Online]. Available: <https://www.intel.cn/content/www/cn/zh/products/sku/236781/intel-core-i7-processor-14700-33m-cache-up-to-5-40-ghz/specifications.html>
- [40] H. Luo *et al.*, “Ramulator 2.0: A modern, modular, and extensible dram simulator,” *IEEE Computer Architecture Letters (RAL)*, vol. 23, no. 1, pp. 112–116, 2023.
- [41] D. Dai *et al.*, “DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024, pp. 1280–1297.
- [42] E. Haider *et al.*, “Phi-3 safety post-training: Aligning language models with a ‘break-fix’ cycle,” *arXiv preprint arXiv:2407.13833*, 2024.