

A Code

Please find the code base for this paper here: <https://anonymous.4open.science/r/mozart-0D75>

B More Experimental Results

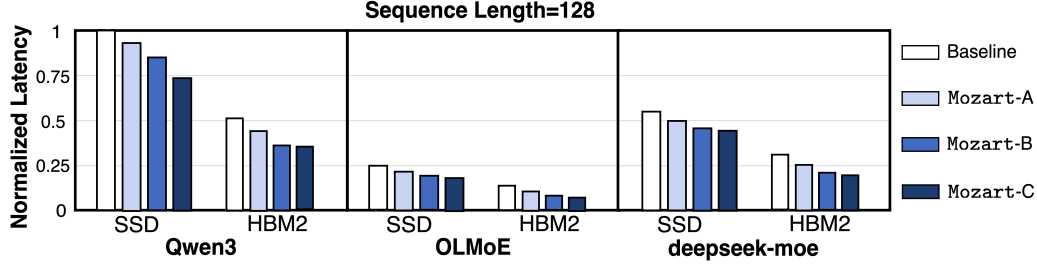


Figure 7: **Normalized Latency Comparison for 3 MoE-LLMs with Sequence length 128.** The max wall-clock latency here is 7.61 s (Qwen3 model with baseline method using SSD for DRAM).

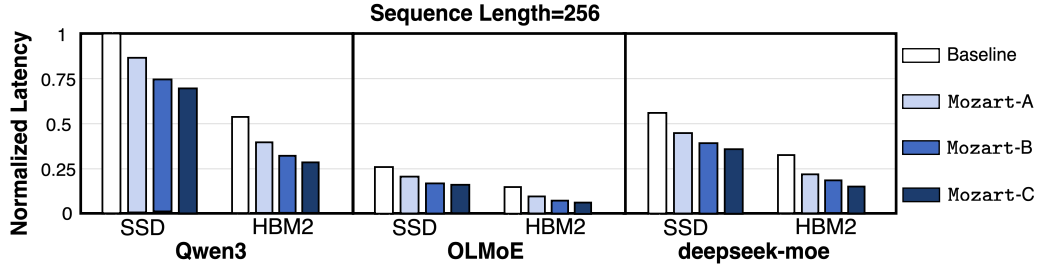


Figure 8: **Normalized Latency Comparison for 3 MoE-LLMs with Sequence length 256.** The max wall-clock latency here is 9.17 s (Qwen3 model with baseline method using SSD for DRAM).

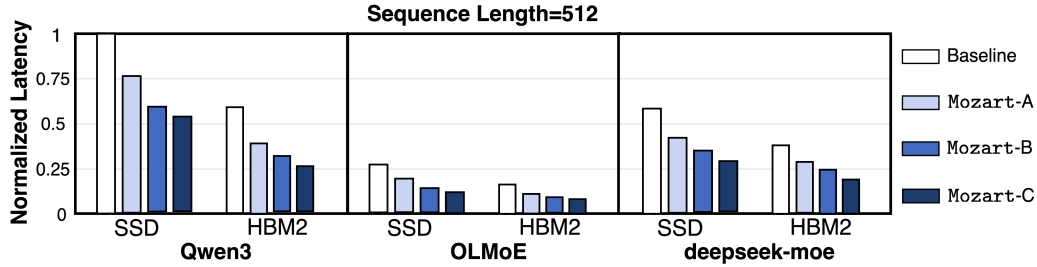


Figure 9: **Normalized Latency Comparison for 3 MoE-LLMs with Sequence length 512.** The max wall-clock latency here is 13.03 s (Qwen3 model with baseline method using SSD for DRAM).

We provide comprehensive numerical latency results for all configurations, including 3 sequence length (128, 256, 512), 4 methods (Mozart Baseline, A, B, and C), and 2 DRAM (SSD and HBM2). Results comparison visualizations are provided in Figure 7, 8, and 9.

C Motivation Explanations

C.1 Why Attention is Memory-Bound and FFN is Compute-Bound

The chiplet architecture in Mozart utilized the fact that in a typical decoder layer in modern large language models, the *Attention* module is memory-bound and the *FFN* module is computation-bound. We demonstrate it using profiling experiments on the OLMo-2 model series [30]. The experiment settings are:

- We examine a single decoder layer, and collect the wall-clock latency and the FLOPs for both attention and FFN modules.

- The results are collected through running the forward pass, *i.e.*, the prefilling stage of model inference, and the results are normalized for easier comparison.
- We fix the batch size to 4 and test the sequence length of 512, 1024, and 2048.
- We select OLMo-2 models with 4 scales, including 1B, 7B, 13B, and 32B.

The profiling experiments are visualized in Figure 10 (1B), Figure 11 (7B), Figure 12 (13B), and Figure 13 (32B). We can find that, the FFN module counts for more FLOPs but less wall-clock latency. It is because the **Attention** module is memory-bound and the **FFN** module is computation-bound:

- The FFN module counts for more FLOPs because it contains more model parameters. But the computation task for it is mainly composed of large matrix multiplication, which is easy to parallelize. Therefore, the wall-clock latency of it can be lower than attention.
- The Attention module requires frequent memory access operations, which is demonstrated by the Flash-Attention series [7, 6, 33]. Although it contains fewer model parameters, the computation tasks here are difficult to parallelize. Therefore, the attention module counts for more wall-clock latency.

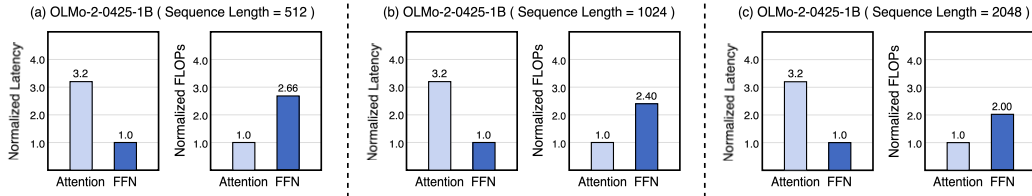


Figure 10: Profiling results on latency & FLOPs for Attention & FFN using OLMo-2-0425-1B.

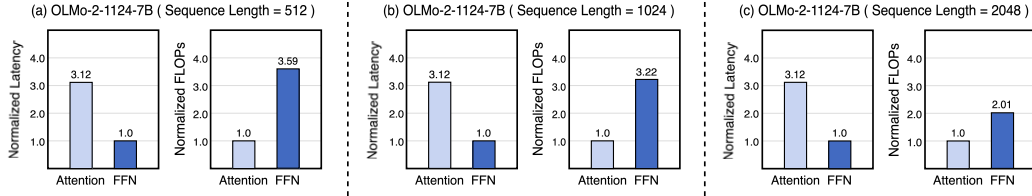


Figure 11: Profiling results on latency & FLOPs for Attention & FFN using OLMo-2-1124-7B.

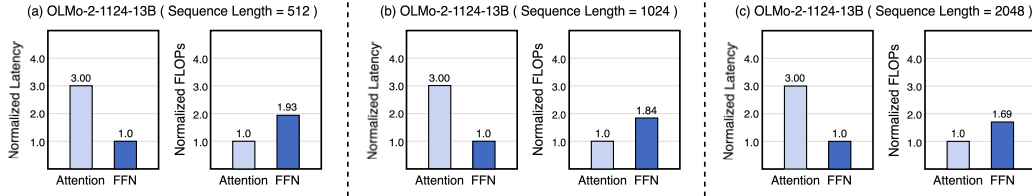


Figure 12: Profiling results on latency & FLOPs for Attention & FFN using OLMo-2-1124-13B.

C.2 Challenges for Mixture-of-Expert Computation

We present 3 challenges for MoE computation in the abstract part of this paper, including memory locality issues, communication overhead, and insufficient computing resource utilization. Our algorithm-hardware co-design scheme in Mozart tries to solve these challenges with joint efforts. We demonstrate these challenges through fine-tuning an OLMoE-1B-7B model with 4-way expert parallelism, with batch size 8 on each GPU and sequence length 512. We use MegaBlocks [13], the standard expert parallelism framework, for the MoE modules, and use data parallelism for the attention modules. We employ the dropless MoE implementation. The training speed is 2-3 iterations per second, and we monitor the behavior of each GPU with an interval of 0.1 s. We take 3 fragments for visualization, as shown in Figure 14, 15, and 16, which demonstrate that both the GPU power and the memory consumption show high dynamism. These phenomena can explain 2 challenges:

- **Memory Locality Issues:** Since the workload for each expert changes dynamically, the activation tensors should be frequently allocated and freed, leading to severe memory management issues.

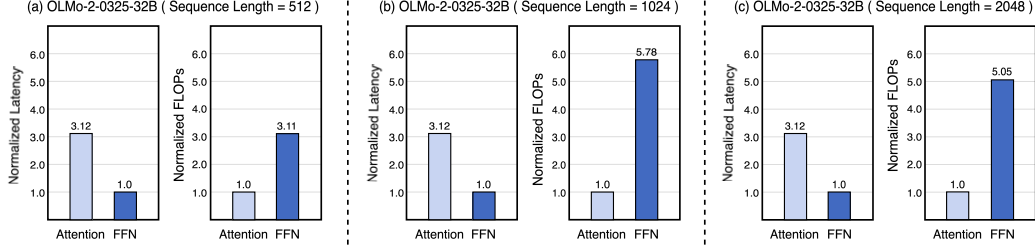


Figure 13: Profiling results on latency & FLOPs for Attention & FFN using OLMo-2-0325-32B.

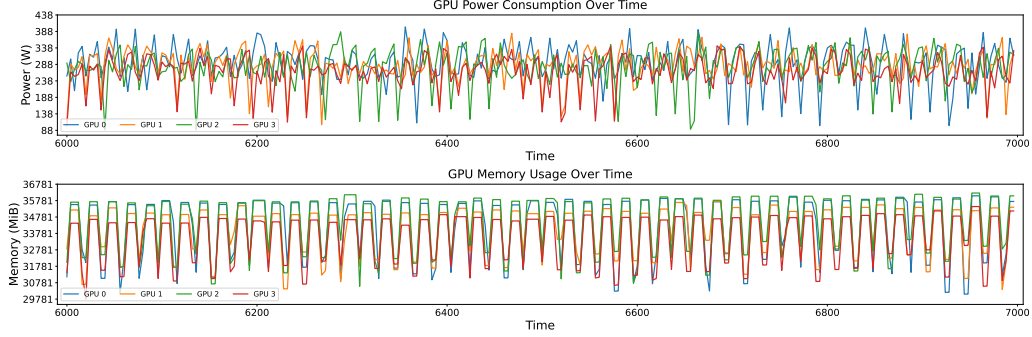


Figure 14: GPU Behavior Monitor at Time Step 6k-7k.

- **Insufficient Computing Resource Utilization:** The reason for this challenge lies in 2 aspects: (1) the dynamism of workload leads to dynamism of GPU power, and (2) the dynamism of workload restricts the training batch size to avoid out-of-memory error, which also constrains the utilization of GPU computing resources.

The all-to-all communication issues have been explained in Tutel [16], which is a significant bottleneck for training MoE models at scale, consuming up to 40% of the total runtime.

D Measuring All-to-All Communication Complexity with $\mathcal{C}_{\mathcal{T}}$

We propose to measure the all-to-all communication data volume in Section 3.3 using the average replication times of each token, denoted as $\mathcal{C}_{\mathcal{T}}$. We prove that $\mathcal{C}_{\mathcal{T}}$ is the least upper bound of the ratio between actual all-to-all communication data volume and the total number of tokens. We take a single all-to-all communication in D -way expert parallelism as an example, and denote the original tokens as $\{\mathcal{S}_i\}_{i=0}^{D-1}$. For a single token $t \in \mathcal{S}_i$ on device i , we denote the number of replications for it transmitting from device i to device j as $N_i^j(t)$, *i.e.*, token t on device i activates $N_i^j(t)$ experts preserved on device j . In the standard expert parallel framework, given top- k routing, we have

$$\sum_{j=0}^{D-1} N_i^j(t) = k, \forall t \in \mathcal{S}_i \text{ and } \forall 0 \leq i \leq D-1. \quad (6)$$

For the actual all-to-all communication data volume:

$$\begin{aligned} \sum_{i=0}^{D-1} \sum_{t \in \mathcal{S}_i} \left(\sum_{j=0}^{i-1} N_i^j(t) + \sum_{j=i+1}^{D-1} N_i^j(t) \right) &\leq \sum_{i=0}^{D-1} \sum_{t \in \mathcal{S}_i} \left(\sum_{j=0}^{i-1} N_i^j(t) + N_i^i(t) + \sum_{j=i+1}^{D-1} N_i^j(t) \right) \\ &= \sum_{i=0}^{D-1} \sum_{t \in \mathcal{S}_i} \left(\sum_{j=0}^{D-1} N_i^j(t) \right) \\ &\leq k \cdot \sum_{i=0}^{D-1} |\mathcal{S}_i| \end{aligned} \quad (7)$$

The 2 inequalities in Equation 7 are reached when

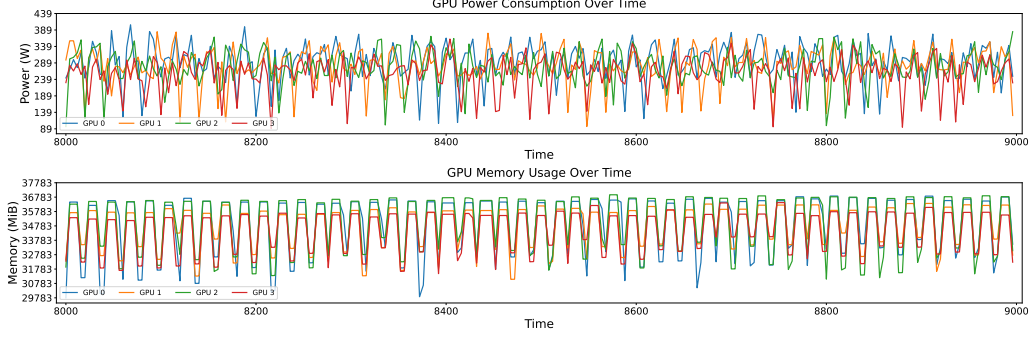


Figure 15: GPU Behavior Monitor at Time Step 8k-9k.

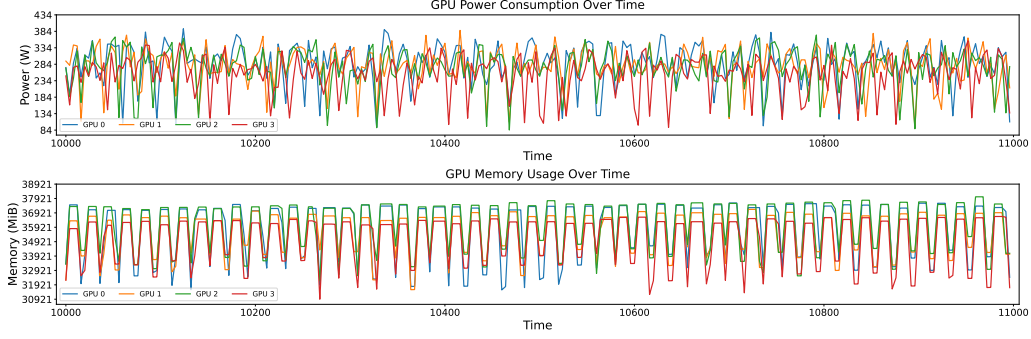


Figure 16: GPU Behavior Monitor at Time Step 10k-11k.

- The first one is achieved when $N_i^i(t) = 0$ for all $0 \leq i \leq D - 1$ and $t \in \mathcal{S}_i$, *i.e.*, no token would activate the experts kept on the device where the token is originally kept.
- The second one is achieved for standard expert parallelism, *i.e.*, making k replications for each token in the dispatch stage under top- k routing.

The first inequality cannot be utilized for communication efficiency, since it is data-dependent and task-dependent. While the second inequality can be leveraged by employing our proposed strategy in Section 3.3.

E Impact Statement

As the paper’s primary innovation is efficiently deploying the post-training process of MoE-based large language models on the chiplet-based system, it by itself doesn’t pose any obvious risks. The potential for negative societal impact depends on the specific MoE-LLMs. We strongly recommend these models be used in compliance with all ethical standards appropriate to the domain in which it is targeted to be deployed.