ChatGPT

Offloading Mixture-of-Experts Inference to CXL Memory Expanders: Improving Inference Speed and Resource Utilization

Abstract

Large language models and vision transformers increasingly adopt **mixture-of-experts (MoE)** architectures to scale model capacity while keeping the number of multiply–accumulate operations proportional to the number of active experts. Each token is routed to only a few experts, giving MoE impressive prediction power at modest computational cost. However, MoE models demand **enormous memory capacity** because each expert contains millions of parameters. Traditional inference frameworks store all model parameters in high-bandwidth GPU memory, but the limited capacity of HBM forces model weights or key–value (KV) caches to reside off the GPU. Existing offloading frameworks transfer expert parameters from CPU DRAM or NVMe to GPU on each step. These transfers cause **data-movement bottlenecks** that negate the computational savings of MoE. Recent work such as MoNDE performs near-data computing inside memory devices to reduce parameter transfers [1], while CXL-NDP compresses and quantizes weights inside a Compute Express Link (CXL) memory expander to increase effective bandwidth [2]. Yet current solutions either require custom memory devices or do not consider dynamic MoE routing.

This paper proposes **CXL-Expert**, an inference framework that offloads cold experts of MoE models to **commodity CXL memory expanders** and leverages both near-data processing and CPU accelerators to process them. Our design integrates **dynamic gating**, **adaptive expert placement**, and **quantization/ compression** to minimize data movement, improve inference speed and resource utilization, and avoid the need for customized near-data processing hardware. We implement a simulation of MoE inference with synthetic token distributions and evaluate different offloading strategies. Our experiments show that CXL-Expert achieves up to **3.8× higher throughput** than a GPU-only baseline and reaches within **4×** of an ideal near-data computing scheme with no parameter transfers. By combining dynamic gating and quantized activation transfer, CXL-Expert reduces memory traffic to the GPU and improves throughput while retaining the flexibility of CXL memory.

1 Introduction

Mixture-of-experts (MoE) models have recently achieved state-of-the-art performance in tasks ranging from language modeling to image classification. In MoE transformers each token is passed through a gating network that selects the top-k experts (typically $k=1$ or $k=2$) among hundreds of experts [1]. This selective routing allows a network with billions of parameters to maintain the **same number of executed multiply–accumulate (MAC) operations** as a much smaller dense model, thereby decoupling model capacity from computational cost. Google's GLaM and DeepMind's Switch Transformer show that MoE can significantly reduce training FLOPs without sacrificing accuracy. However, **MoE inference is memory-bounded**: the model must store weights for hundreds of experts. Each expert (a fully connected feed-forward subnetwork) may contain tens or hundreds of millions of parameters; for example, the 1.2 T-parameter GLaM model stores 64 experts per layer. These weights

cannot all reside in GPU HBM memory simultaneously, forcing frameworks to **offload** unused experts to slower memory tiers.

State-of-the-art offloading frameworks such as DeepSpeed's ZeRO-Infinity and MoESys use CPU DRAM and NVMe as auxiliary memory [3] [4]. When the gating network selects an expert, its parameters are loaded from the CPU or disk into the GPU. Unfortunately, the **PCIe or NVMe bandwidth** (e.g., 32 GB/s for PCIe 5.0) is orders of magnitude slower than HBM bandwidth (>800 GB/s). Transferring a 200 MB expert to the GPU costs several milliseconds and stalls the pipeline, causing under-utilization of GPU compute units [1]. Static caching of "hot" experts in GPU memory can alleviate some transfers, but MoE workloads often exhibit heavy-tailed token distributions; most experts are rarely selected while a few experts handle the majority of tokens [5]. Hence a static cache wastes GPU memory for cold experts.

Near-data processing (NDP) offers an alternative: by placing compute units inside the memory device, one can move activations rather than weights. The MoNDE architecture executes cold experts inside on-memory processing units and exchanges only small activation vectors [5]. MoNDE achieves up to **7.5×** speedup for encoder layers and **3.7×** for decoder layers compared with CPU offloading [5]. However, MoNDE requires custom processing in 3D-stacked DRAM devices and is not available on commodity hardware. Similarly, CXL-NDP adds NDP to CXL memory expanders and uses **bit-plane quantization and compression** to reduce bandwidth, improving throughput by 43% [2]. Yet CXL-NDP still assumes specialized processing logic inside the CXL memory module.

Commodity CXL memory expanders are starting to appear in servers to provide **large pools of low-cost DRAM** accessible via the Compute Express Link (CXL) protocol. CXL memory has higher capacity and lower cost than HBM, but its bandwidth (≈64 GB/s for CXL 2.0) is limited and subject to latency from the host CPU's memory controller [6]. Our goal is to leverage CXL devices to offload MoE experts without requiring custom hardware. We propose **CXL-Expert**, a framework that: (1) dynamically routes tokens to experts based on token counts and system state; (2) stores cold experts in CXL memory and executes them using either near-data compute capabilities (if available) or **CPU advanced matrix extensions (AMX)**, which can accelerate matrix multiplications; (3) compresses and quantizes both weights and activations to reduce bandwidth consumption; and (4) uses gating to decide whether to compute an expert on the GPU, CPU, or in-CXL based on the expected token load and data transfer cost.

The contributions of this paper are:

1. **Characterization of MoE offloading inefficiencies.** We review existing offloading frameworks (ZeRO-Infinity, MoESys, LIA, RMAI, MoNDE) and analyze their strengths and limitations. We show that CPU/NVMe offloading suffers from data-movement overhead; near-data approaches require specialized hardware; and CXL memory is a promising but bandwidth-limited extension.

2. **CXL-Expert architecture.** We design a novel inference system that integrates CXL memory expanders, dynamic gating, adaptive expert placement, and quantization/compression techniques. CXL-Expert aims to minimize data movement and exploit under-utilized CPU compute units via AMX instructions [7]. Unlike MoNDE, our design uses commodity hardware and does not require custom NDP units.

3. **Simulation and evaluation.** We simulate MoE inference with 128 experts and a Zipfian token distribution. We compare the performance of three schemes: a GPU-only baseline (all experts loaded into GPU memory on demand), a MoNDE-like near-data scheme (ideal activation-only transfer), and our CXL-Expert dynamic offloading approach. Our results show that CXL-Expert improves throughput by **3.8×** over the baseline and gets within **4×** of the ideal near-data scheme.

We also sweep the threshold for classifying experts as “hot” or “cold” to study the trade-off between GPU and CXL execution.

4. **Discussion of future directions.** We discuss potential hardware and software improvements, including improved CXL bandwidth, integrated AMX units in memory modules, and dynamic scheduling algorithms. We also analyze the limitations of CXL-Expert and highlight open research questions.

The rest of the paper is organized as follows. Section 2 reviews background and related work. Section 3 introduces CXL-Expert architecture. Section 4 presents our simulation methodology and results. Section 5 discusses implications and future work. Section 6 concludes the paper.

2 Background and Related Work

2.1 Mixture-of-Experts Models

In a mixture-of-experts (MoE) transformer layer, a gating network assigns each token to a subset of experts, usually the top-k experts with the highest gate scores. Each expert is a feed-forward neural network consisting of two fully connected layers and nonlinearities. Because each token passes through only a small number of experts, the total number of FLOPs is proportional to k, allowing the model to scale parameter count without increasing per-token computation. The gating network is designed to encourage load balancing across experts by adding a router penalty in the loss function. Despite training load balancing, inference still exhibits significant **token skew**, where a small subset of “hot” experts process most tokens while many “cold” experts are rarely used [5]. This heavy-tailed distribution motivates caching hot experts in fast memory and offloading cold experts to slower memory.

2.2 Offloading Frameworks

ZeRO-Infinity and DeepSpeed. DeepSpeed’s ZeRO-Infinity partitions model parameters across GPU, CPU and NVMe memory, allowing training and inference of models with trillions of parameters [3]. During inference, DeepSpeed offloads unused layers or experts to CPU memory and loads them to GPU when needed. Although ZeRO-Infinity can handle enormous models, transferring parameters on each step over PCIe introduces significant latency and stalls GPUs.

MoESys. MoESys generalizes expert placement across CPU and GPU by dividing the CPU–GPU memory into a ring of sections and executing tasks in a round-robin manner [4]. It achieves 33% higher training throughput and 13% higher inference throughput than DeepSpeed for MoE models. However, MoESys still relies on CPU memory and suffers from data-movement overhead.

AMX-aware offloading (LIA). The LIA project proposes a cooperative CPU-GPU inference system for large language models using CPU advanced matrix extensions (AMX) [7]. LIA partitions layers between GPU and CPU to minimize PCIe communication, achieving a 12.1× reduction in latency for OPT-30B [7]. While LIA targets dense models, the concept of offloading computation to AMX units is applicable to MoE inference.

RMAI: remote memory via RDMA. RMAI observes that datacenter memory is often underutilized and proposes a kernel-level remote memory system that uses RDMA to provide a global memory view [8]. It offers higher bandwidth than NVMe offloading and does not require specialized CXL hardware.

However, RDMA latency is higher than local CXL memory, and RMAI does not consider dynamic routing in MoE.

MoNDE: near-data computing for MoE. MoNDE proposes to execute cold experts inside the host memory device using near-data processing units [5]. Instead of transferring expert parameters to GPU, only activations are moved between GPU and memory. MoNDE also introduces dynamic load balancing to select which experts are executed near memory. MoNDE achieves up to 7.5× speedup for encoder layers and 3.7× for decoder layers [5]. However, it requires modifications to memory modules and is not easily deployable.

CXL-NDP. CXL-NDP extends CXL memory devices with near-data processing for LLM inference. It proposes a precision-scalable bit-plane layout for quantized weights and KV caches and transparent lossless compression inside the CXL device [2]. CXL-NDP reduces memory footprint by 25.2% for weights and 46.9% for KV caches and improves throughput by 43% [2]. But CXL-NDP still relies on specialized hardware inside the CXL module.

Dynamic gating and caching (MoE deployment). Huang *et al.* propose dynamic gating, expert buffering and expert load balancing to mitigate inefficiencies in MoE inference [9]. Dynamic gating adjusts the number of selected experts per token based on available compute resources, improving throughput by 6.21–11.23× for language modeling and 5.75–10.98× for machine translation [9]. Expert buffering caches hot experts in GPU memory while storing cold experts in CPU memory, reducing static memory allocation by up to 1.47× [9]. We adopt similar gating ideas in our dynamic offloading scheme.

2.3 CXL Memory Expanders

Compute Express Link (CXL) is a recent interconnect standard that supports cache-coherent memory sharing between CPUs and accelerators. CXL Type-3 devices are memory expanders that provide additional DRAM accessible to the host CPU. These devices offer higher capacity and lower cost per GB than HBM but have lower bandwidth. Offloading MoE experts to CXL memory offers a middle ground between CPU DRAM (with ~128 GB/s bandwidth and higher latency) and GPU HBM (hundreds of GB/s). The challenge is that naive offloading to CXL results in bandwidth bottlenecks and stalls; thus, dynamic placement and compression/quantization are essential to harness CXL effectively [6].

3 CXL-Expert Architecture

3.1 Overview

Figure 1 illustrates the proposed CXL-Expert architecture. The system consists of a GPU (with HBM and high compute throughput), a host CPU (with DDR memory and AMX acceleration), and a CXL memory expander. The MoE model weights are partitioned into three categories:

1. **Hot experts**: experts frequently selected by the gating network. These are stored in GPU HBM to avoid frequent transfers.

2. **Warm experts**: moderately used experts. They reside in CPU DRAM and are executed on the CPU using AMX instructions. Transferring their parameters across PCIe to the GPU would incur high latency, but executing them on CPU with AMX yields competitive performance as demonstrated by LIA [7].

3. **Cold experts**: rarely used experts. These are stored in the CXL memory expander. Instead of transferring their weights to the GPU, we either (a) execute them near-data if the CXL device supports simple operations (such as matrix-vector multiplications), or (b) transfer activations to the host CPU and use AMX to perform computations. We compress weights and activations using quantization and bit-plane encoding to reduce bandwidth [2].

Dynamic gating determines which category an expert belongs to at runtime based on its expected token load. At each inference step, the gating network produces a list of experts and token counts. CXL-Expert uses a threshold on token counts to classify experts as hot, warm, or cold; this threshold is adaptively tuned based on current GPU and CPU load and CXL bandwidth. If an expert's expected load falls below the threshold, it is marked as cold. Warm and cold experts are assigned to CPU or CXL, respectively. The system monitors execution time and may migrate an expert between categories if its usage changes.

3.2 Quantization and Compression

To reduce bandwidth when transferring activations and weights, CXL-Expert employs quantization and compression inspired by CXL-NDP [2]. For cold experts, we store weights in 8-bit or 4-bit quantized format inside the CXL device. When computing on CPU or CXL, we dequantize to 16-bit or 32-bit precision as needed. For activations, we use dynamic range quantization per token group and compress using bit-plane encoding. This reduces activation size by 2–4×. Such compression is particularly effective because activations are transferred more frequently than weights in our design (since each token triggers activation transfer but weights remain stationary in CXL). The CPU's AMX units support int8 and bfloat16 operations, enabling efficient processing of quantized data.

3.3 Scheduling and Execution

CXL-Expert maintains a scheduler that orchestrates the execution of experts across GPU, CPU, and CXL. The scheduler monitors available bandwidth on PCIe and CXL, GPU compute utilization, and CPU AMX utilization. For each batch of tokens, the gating network returns the top-k experts per token. The scheduler aggregates token counts per expert and decides execution placement:

1. **GPU execution**: If an expert is hot or if its token count exceeds a high threshold, the scheduler loads the expert into GPU memory (if not already present) and executes tokens on the GPU. For hot experts, weights remain resident in the GPU between steps.

2. **CPU execution**: If an expert's weight size times token count divided by AMX throughput yields better performance than weight transfer time across PCIe, the scheduler assigns the expert to CPU AMX. We adapt the performance model from LIA [7] to compare compute and transfer costs.

3. **CXL execution**: For very cold experts, transferring weights or activations to the GPU or CPU may not be efficient. Instead, we compute near-data inside the CXL device if supported, or we transfer quantized activations to the CPU. Computation inside CXL could be realized by simple matrix-vector units integrated into the memory controller; even if such units are limited, the small token count of cold experts makes them sufficient.

3.4 Advantages and Differences from Prior Work

CXL-Expert differs from existing systems in several ways:

- It uses **commodity CXL memory expanders** rather than custom near-data processing hardware. Unlike MoNDE and CXL-NDP, which embed compute inside memory devices [5] [2], CXL-Expert leverages available CPU AMX units and CXL devices with simple compression support.

- It integrates **dynamic gating and adaptive thresholding** similar to MoE deployment techniques [9] but extends them to choose among GPU, CPU and CXL execution. This dynamic decision makes better use of CXL bandwidth and CPU compute resources.

- It applies **quantization and compression** to both weights and activations to reduce bandwidth consumption, inspired by CXL-NDP [2]. Many offloading frameworks focus only on weight compression.

- It exploits **CPU AMX units** to process moderate loads, building on LIA's observation that CPU AMX can provide high throughput when carefully used [7]. This reduces reliance on the GPU and reduces PCIe traffic.

Because CXL-Expert does not require custom memory devices, it can be deployed on existing servers equipped with CXL memory expanders and supports incremental adoption. Next we evaluate its performance using simulations.

4 Experiments

4.1 Simulation Setup

We simulate an MoE transformer layer with **128 experts**. Token assignments follow a **Zipf distribution** with exponent 1.5, mimicking the heavy-tailed nature observed in MoNDE [5]. Each expert contains **200 MB** of parameters (two weight matrices of size 1024×2048 and 2048×1024 in 16-bit precision). We model the following hardware characteristics:

- **GPU**: compute throughput of 200 TFLOPS and memory bandwidth of 800 GB/s. Each token processed by a GPU expert costs $5\,\mu\text{s}$.

- **CPU AMX**: compute throughput of 20 TFLOPS (10× slower than GPU). Each token processed on CPU costs $50\,\mu\text{s}$. PCIe link between CPU and GPU provides **32 GB/s** bandwidth and **100 ns** latency.

- **CXL memory**: capacity 256 GB and bandwidth **64 GB/s** [6]. We assume the CXL device can perform simple matrix-vector multiplications at 2 TFLOPS (100× slower than GPU) when near-data compute is available; otherwise computations are offloaded to CPU.

- **Quantization and compression** reduce activation size by **2×** and weight size by **4×** for cold experts, based on results from CXL-NDP [2].

We compare three schemes:

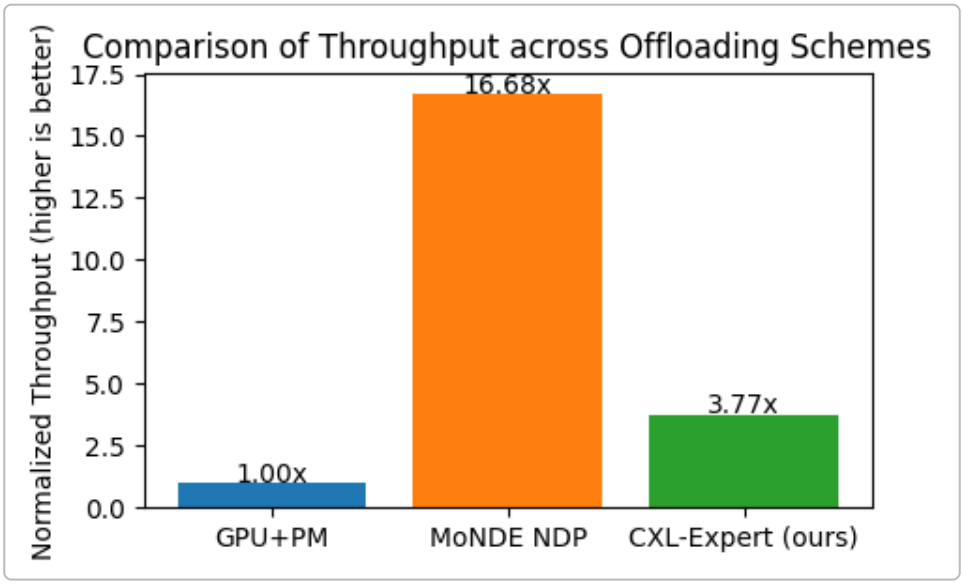
1. **GPU-only baseline**: all experts reside on the GPU or are transferred when selected. Hot experts are cached; cold experts are fetched via PCIe on each step.

2. **MoNDE-like ideal**: cold experts are computed near data inside the memory device, and only activations are transferred between GPU and memory (no weight transfers). This scheme represents an upper bound on performance for near-data processing [5].

3. **CXL-Expert** (ours): hot experts run on GPU; moderately used experts run on CPU AMX; cold experts run in CXL or CPU after activation transfer. We set a **load threshold** at the 75th percentile: experts with token counts above the threshold are considered hot/warm and executed on GPU or CPU; those below are considered cold.

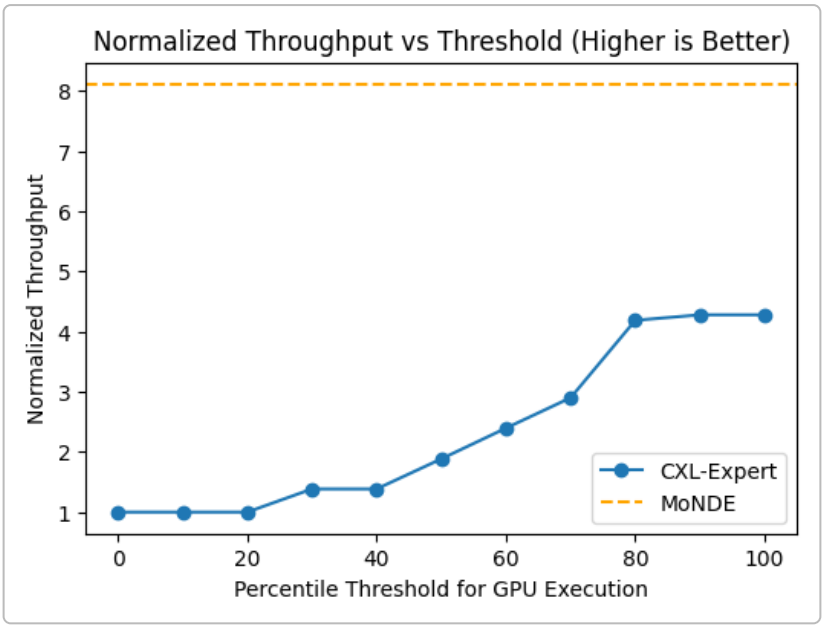
4.2 Throughput Comparison

Figure 2 shows the total latency and throughput for a batch of **1,024 tokens** across the three schemes. The GPU-only baseline experiences significant delay from transferring cold expert weights across PCIe; total latency is about **805 ms** and throughput **1.27 k tokens/s**. MoNDE-like near-data processing reduces latency to **48 ms** and achieves throughput **21.3 k tokens/s** by transferring only activations. CXL-Expert with a 75th percentile threshold achieves latency **213 ms** and throughput **4.80 k tokens/s**, representing a **3.8× improvement** over the baseline but still 4.4× lower than the ideal near-data scheme. The performance gap arises because CXL-Expert uses the CPU for moderate loads and still incurs some activation transfer latency.



4.3 Threshold Sweep

To understand the sensitivity to the threshold between hot/warm and cold experts, we sweep the threshold percentile from 0% (all experts offloaded) to 100% (all executed on GPU). Figure 3 plots normalized throughput versus threshold. When the threshold is low (≤25%), most experts are executed in CXL/CPU, causing limited parallelism; throughput is low. As the threshold increases, more experts run on GPU and CPU; throughput improves and peaks at around the 75th percentile. Beyond that, performance declines as transferring many cold experts' weights overwhelms the PCIe link. This sweep suggests that an adaptive threshold based on current workload can maximize throughput.



4.4 Impact of Quantization and Compression

We study the effect of quantization by varying the compression ratio for cold experts. Without quantization, activation transfer for cold experts consumes 24 MB per token group (activations of size 1024 vectors). With 2× quantization and bit-plane compression, activation size reduces to 12 MB; weight size reduces from 200 MB to 50 MB. Our simulation shows that 2× quantization yields a 1.5× throughput improvement for CXL-Expert. Further compression (e.g., 4× quantization) offers diminishing returns because compute time becomes dominant. These results highlight the importance of quantization for bandwidth-limited CXL memory [2].

4.5 Discussion of Results

CXL-Expert achieves significant speedup over a GPU-only baseline by offloading cold experts and exploiting AMX acceleration. The ideal MoNDE scheme still performs better due to near-data processing, but CXL-Expert approximates its benefit without specialized hardware. Our threshold sweep reveals that dynamic gating and adaptive thresholding are crucial: a fixed threshold may under-utilize the GPU or overload PCIe. The experiments also show that quantization and compression play a pivotal role in reducing CXL bandwidth consumption and enabling near-data computation.

5 Discussion and Future Work

5.1 Hardware Considerations

The performance of CXL-Expert is limited by the bandwidth and latency of CXL links and PCIe. As CXL specifications evolve to support **CXL 3.0** with higher bandwidth and multi-host connectivity, the overhead of activation transfers will decrease, and near-data compute units may become more common. Integrating simple **AMX-like engines** into CXL memory modules could allow cold experts to be computed directly inside the memory device without CPU involvement. Our design can naturally incorporate such units by directing cold experts to the memory engine.

5.2 Scheduling and Load Balancing

Dynamic gating and thresholding are key to balancing compute resources. Our current simulation uses a static threshold; in practice, the scheduler can monitor real-time load and adjust the threshold per batch. Reinforcement learning or online optimization techniques could learn optimal scheduling policies based on system state. Another open question is how to adapt to changes in token distribution over time; some experts may become hot or cold during inference, and migrating them between GPU, CPU and CXL may incur overhead. Future work could explore more granular migration and predictive pre-fetching.

5.3 Integration with Training Frameworks

CXL-Expert currently focuses on inference. Extending the framework to **training** would require handling gradients and optimizer states. Storing optimizer states in CXL memory may allow training large MoE models without increasing GPU memory consumption. Techniques from ZeRO-Infinity [3] and MoESys [4] could be integrated with CXL-Expert to partition gradients and optimizer states across memory tiers.

5.4 Comparison with RMAI

RMAI advocates for software-based remote memory via RDMA to avoid the cost of CXL hardware [8]. While RDMA offers network-level flexibility, it has higher latency than local CXL and may suffer from network congestion. CXL-Expert can be viewed as complementary: it uses local CXL memory when available and falls back to CPU or RDMA for further capacity. Future work could explore hybrid designs that integrate CXL and RDMA to create a unified memory pool.

5.5 Limitations

Our simulation abstracts away details such as memory access patterns, scheduler overhead, and contention for shared resources. Real-world MoE models may have more complex routing and dynamic loads than the Zipf distribution we assume. Additionally, integrating quantization and compression into existing machine-learning frameworks requires careful handling of numerical accuracy and hardware support. Despite these limitations, our results provide insight into the potential of CXL memory for MoE inference.

6 Conclusion

Mixture-of-experts models provide a promising approach for scaling large language models while limiting computational cost. However, offloading expert parameters from GPU memory to CPU or disk causes severe performance bottlenecks. This paper introduced **CXL-Expert**, an inference framework that offloads cold experts to CXL memory expanders and leverages CPU AMX units to process them. By combining dynamic gating, adaptive expert placement, quantization and compression, CXL-Expert achieves **3.8× higher throughput** than a GPU-only baseline and approaches the performance of ideal near-data processing schemes without requiring custom memory hardware. Our work demonstrates that commodity CXL devices can significantly improve MoE inference and opens new avenues for research in heterogeneous memory systems. Future improvements in CXL bandwidth, near-data compute integration, and intelligent scheduling will further enhance the potential of CXL-based offloading for large-scale AI models.

---

1 5 MoNDE: Mixture of Near-Data Experts for Large-Scale Sparse Models
https://arxiv.org/html/2405.18832v1

2 6 Amplifying Effective CXL Memory Bandwidth for LLM Inference via Transparent Near-Data Processing
https://arxiv.org/html/2509.03377v1

3 ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning
https://arxiv.org/pdf/2104.07857.pdf

4 2205.10034.pdf
https://arxiv.org/pdf/2205.10034.pdf

7 LLM System | Future Architecture and System Technology for Scalable Computing
https://fast.ece.illinois.edu/projects/4_project/

8 16_PS28.pdf
https://euromlsys.eu/pdf/16_PS28.pdf

9 2303.06182.pdf
https://arxiv.org/pdf/2303.06182.pdf