Marissa Bhavsar (mbhavsar), Shaurya Gunderia (shagund), Christopher Lin (cwlinn), Raghav Ramesh (rmramesh)

Summary of *dLoRA: Dynamically Orchestrating Requests and Adapters for LoRA LLM Serving*

Problem and Motivation

The demand for fine-tuning large language models (LLMs) on specific domains has increased, and Low-Rank Adaptation (LoRA) presents a parameter-efficient solution that reduces fine-tuning costs by focusing on a small subset of model parameters. However, serving multiple LoRA-adapted models from a single pre-trained LLM base model creates unique challenges in inference. Current LLM inference systems (such as Orca, vLLM, and HuggingFace's TGI) are optimized for single models, resulting in inefficient memory use and poor load distribution when applied to multiple LoRA adapters. Specifically, the need to serve various LoRA-adapted versions of a base LLM introduces issues like low GPU utilization and high memory footprint, as well as load imbalances due to the variable lengths of input and output across requests. These inefficiencies significantly limit the scalability, latency, and throughput of serving models with multiple LoRA adapters. dLoRA addresses these issues by dynamically managing both request batching within replicas and the migration of LoRA adapters across replicas to optimize memory and resource usage.

Related Works

The dLoRA paper builds on existing work in LLM inference optimization, specifically on systems like vLLM and HuggingFace TGI. These systems focus on single-LLM inference and lack efficient handling of multi-adapter scenarios. The vLLM framework does not account for load distribution across multiple replicas with variable-length requests and also does not support concurrent LoRA models. HuggingFace TGI lacks dynamic batching and cluster-level load management. This can lead to latency and memory inefficiencies. S-LoRA and Punica are concurrent efforts for serving multiple LoRA LLMs by batching requests destined for different adapters. Neither S-LoRA or Punica consider dynamic load balancing or request batching as proposed by dLoRA.

Solution Overview

As mentioned above, there are two main challenges that are present in current LLM inference systems. First, when dealing with merged inference, we experience low GPU utilization within replicas when we receive diverse requests. Second, because input and output lengths are highly variable, severe load imbalancing occurs between different worker replicas. dLoRA aims to address the problems of current LLM inference systems by (1) dynamically merging or unmerging LoRA adapters with the base model within replicas and (2) by migrating LoRA adapters across replicas to balance the load. Each of these solutions present their own challenges. First, it becomes a challenge to determine when to switch between merged and unmerged states.

Second, it becomes a challenge to decide which requests and adapters are migrated, while also keeping the dependency between requests and adapters in mind. The design proposes to address these two challenges respectively with the following two techniques: (1) Dynamic Cross-Adapter Batching and (2) Request-Adapter Co-Migration.

Dynamic Cross-Adapter Batching in dLoRA operates within each worker replica (intra-replica) to optimize GPU utilization by dynamically switching between merged and unmerged inference modes. In merged inference, the LoRA adapter is combined with the base LLM weights, eliminating additional computational overhead and making it ideal for handling high volumes of similar requests by reducing repeated computations. However, merged inference can create queuing delays when requests involve multiple adapters. In contrast, unmerged inference keeps the base model and adapters separate, enabling parallel processing of various adapter layers within a single batch to maximize batch size and reduce queuing delays for diverse requests, though it comes with additional computational overhead from separate matrix operations. The choice between these modes is guided by a dynamic batching algorithm that evaluates the workload and request patterns at runtime to determine the most efficient mode, using adaptive thresholds (αswitch and βswitch) tuned based on historical data to balance batch size and switching costs. Additionally, dLoRA's credit-based mechanism ensures fairness by prioritizing less frequent requests when their credits reach a threshold, preventing starvation. This combined approach, considering execution time, queuing delay, and switching overhead, ultimately aims to reduce end-to-end latency and maintain balanced processing across adapters.

Request-Adapter Co-Migration takes place across replicas (inter-replica). To prevent load imbalances, proactive and reactive mechanisms take place. First, proactive load balancing handles predictable workload patterns by distributing requests evenly across replicas initially. dLoRA utilizes a dispatcher that assigns requests to replicas based on known patterns (like peak and off-peak times). dLoRA relies on adapter-aware dispatching, where it considers the load of each adapter across replicas to maximize the overall system's burst tolerance. Reactive load balancing then manages unpredictable variations in workload patterns. It then uses the request-adapter co-migration algorithm (modeled as an Integer Linear Programming (ILP) problem) to redistribute both requests and adapters across replicas in real time. By balancing GPU memory demands and processing loads, this approach minimizes delays caused by queuing and ensures efficient GPU utilization.

Limitations

(1) The use of ILP for request-adapter co-migration is computationally expensive, particularly as cluster size grows. Although dLoRA reduces the frequency of migration to mitigate this, the ILP formulation may limit scalability in high-demand environments.

(2) dLoRA relies on dynamic batching based on request types and arrival rates, making it sensitive to unpredictable workload shifts. Sudden bursts of requests with a particular adapter type or length variability may impact latency due to switching or migration delays. This is masked by the paper by just using the average inference time, as discussed in the presentation.

Future Research Directions

The paper does not explicitly discuss future work, but we have provided some potential research directions based on identified limitations presented in the paper.
(1) Optimization of Load Balancing Techniques: To address the computational intensity of ILP, future research could explore heuristic or machine learning-based approaches for load balancing that would retain the benefits of dynamic migration but reduce the need for frequent ILP computation.
(2) Integration with Cloud-Based Inference Systems: Adapting dLoRA for cloud-based LLM inference serving on platforms like AWS, Azure, or Google Cloud, where resources are distributed and virtualized, could open up new applications and require new orchestration techniques for efficient cross-node communication and resource sharing.

Summary of Class Discussion

Why is transferring LoRA adapters necessary?: Transferring LoRA adapters across replicas helps dLoRA dynamically balance workloads, allowing reused adapters to speed up response times on servers where they are frequently needed, although this requires some migration and load balancing.

Is there a sense of fairness in adapter handling?: dLoRA doesn't enforce strict fairness but uses a credit-based mechanism to prevent starvation, allowing less-used adapters to "earn" credits and be processed periodically.

How are adapters stored, and why does each server seem to handle only one replica at a time?: The base model is stored on the GPU, while adapter components (A and B matrices) reside in CPU DRAM. Specific adapters are often used on certain servers, creating the appearance of exclusive replica handling due to historical performance trends.

What load balancing techniques are employed?: dLoRA uses proactive load balancing to anticipate high-demand adapters based on historical data, and reactive balancing to redistribute requests as demand spikes occur in real time.

Are there concerns with average latency as a performance metric?: Average latency is helpful but may be too vague; additional metrics like queuing time or memory fragmentation could offer a more complete view of efficiency.

Summary of *Mixture of LoRA Experts*

Problem and Motivation

LoRA, Low-Rank Adaptation, is a fine tuning method for LLMs that is resource-friendly and efficient. LoRA uses low rank decomposition instead of modifying the entire weight matrices. However, one issue with LoRA is that it is generally task specific. LoRA is modular, so multiple LoRA modules can be added to a single model to handle multiple tasks. This makes it versatile and scalable, as these small layers can be switched in and out without needing to re-train the full model. However, adapting LoRA for a wide range of tasks remains a challenge, as task diversity may exceed the model's capacity. In addition, current prevailing methods compose the trained LoRAs directly, which can have the downside of erasing the unique characteristics of individual trained LoRAs. To address these issues, MoLE seeks to compose multiple trained LoRAs dynamically and efficiently, without losing their individual characteristics through a mixture of experts.

Related Works

In discussing MoLE, the paper also refers to many other related works that have employed similar techniques that either differ to those of MoLE or that are improved upon with the author's solution. For the LoRA composition methods, the paper mainly discusses linear arithmetic composition from "Composing parameter-efficient modules with arithmetic operations" (Zhang et al., 2023; Huang et al., 2023; Han et al., 2023) and reference tuning-based composition from "Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models" (Gu et al., 2023). The paper first mainly discusses two different LoRA composition methods that have been developed in the related works mentioned above but fall short in providing the efficiency that MoLE achieves. For example, linear arithmetic composition directly combines multiple LoRAs but results in a negative impact to the pre-trained model's generative performance. To reduce this effect, weight normalization is used but may erase the uniqueness of the individual LoRAs. On the other hand, reference tuning-based composition achieves better performance, but lacks flexibility and requires a full model to be retrained. Following the discussion of each of these works, the paper then highlights the need to dynamically and efficiently compose multiple LoRAs while preserving all of their individual characteristics.

Solution Overview

The MoLE paper presents a solution designed to enhance the flexibility and efficiency of large language models by managing multiple LoRA models. The goal of the authors was to retain the uniqueness of each model while also allowing the model to generalize across tasks. The framework introduces a hierarchical weight control system allowing the model to use each layer of trained LoRAs as an 'expert' with a learnable gating function in each layer. This gating

function essentially learns the optimal composition weights for all the LoRAs such that the model can dynamically assign tasks to the most suitable experts having the highest weight for the domain objective. This in turn allows MoLE to dynamically enhance the desirable characteristics while mitigating less favorable ones from each LoRA. With the hierarchical control layer, MoLE is able to organize multiple LoRA experts so that branching decisions can be made at different levels. This also allows the model to select only the necessary experts and avoid an increase in computational load and resource usage. Unlike the related works mentioned previously, MoLE does not combine all LoRAs together and instead leverages the gating function to activate only the experts relevant to a given input or task similar to the mixture of experts technique that this solution employs. An additional benefit of using a mixture of LoRA experts is MoLE can basically mask unwanted LoRAs by the distribution of weight values avoiding the need for retraining the experts and allowing for flexible adaptation to different scenarios.

As can be seen from the proposed solution, the key insight guiding this solution stems from the observations about LoRA behavior and limitations of existing composition methods. To address the limitations of the linear arithmetic composition, MoLE consists of separation between LoRA models ensuring the preservation of the individual characteristics of trained LoRAs and avoids the limitations of the linear arithmetic composition. Additionally the branching mechanism used by MoLE is also adaptable enough to apply different LoRA experts based on input data as shown by its effectiveness across Natural Language Processing (NLP) and Vision & Language (V&L) tasks. This demonstrates improved flexibility and avoids the retraining of models both of which address the limitations of the reference tuning-based composition.

Limitations

MoLE faces the following challenges:

(1). Although MoLE outperforms the other LoRA composition methods, its performance declines when the number of LoRAs being used increases to a large number, for example 128. This highlights an area for MoLE to be improved upon to be able to handle large-scale LoRA composition.

(2) Despite the paper mentioning that block-wise and layer-wise gating granularities achieved the highest performance, and matrix-wise and network-wise granularities underforming, it still leaves room for determining the most optimal granularity to maximize the efficiency of MoLE.

Future Research Directions

Future research directions for Mixture of LoRA Experts (MoLE) could explore several promising areas. First, improving the selection mechanism for expert activation—either by enhancing the efficiency of gating networks or by using more adaptive approaches based on

input context—could lead to better performance in diverse tasks. This can also lead to further balancing between the two losses (domain and balancing). Additionally, experimenting with dynamic routing mechanisms or hierarchical gating models might enable more nuanced allocation of experts, allowing the model to scale effectively across even larger tasks or domains. One could also look into integrating MoLE with more complex multi-modal frameworks, investigating how LoRA adaptations can handle both textual and visual information simultaneously. Another important area could involve fine-tuning MoLE for better interpretability and alignment with specific applications, such as healthcare or finance, where explainability is crucial.

Summary of Class Discussion

- Is there accuracy loss from a request for one lora, but the gaining function distributes it to more than 1?
    - The domain specific loss always takes priority over the balancing loss. In this case, only the specific lora would be chosen.
- The E Omega parameter in the gaining function is based on the sequence length, could you go more into detail on it?
    - Sequence length effects gaining function by dictating which lora adapters are mapped to that specific token in the sequence length.
- Is there anything for the experts to limit how much they dominate the parameters of the base model?
    - If nothing in the sequence is indicative of any adapters in the model, then it will default to the base parameters.
- How are the ground truth values for images created in order to measure the accuracy of the image generation?
    - The authors used ChatGPT to generate the ground truth images
- How is the training data constructed? For the case of needing multiple experts?
    - Not that much detail on how the ground truth images. Probably used ChatGPT to generate these images.