UDC 004.8

Oleksandr Korovii

TRAINING LARGE LANGUAGE MODELS ON HPC GPU CLUSTER: AN OVERVIEW

Training large language models (LLMs) on HPC clusters presents significant challenges, including efficient memory management, parallelism, and minimizing communication overhead. This paper compares three methods – Megatron-LM, ZeRO, and PyTorch Fully Sharded Data Parallel (FSDP) – highlighting their approaches and effectiveness in addressing these challenges. The analysis provides insights into the advantages and limitations of each method, offering guidance for optimizing LLM training on large-scale HPC systems.

Keywords: Large Language Models, GPU Cluster, Distributed Training Methods.

Relevance of the Research Topic. The rise of large language models (LLMs) like GPT-3 and BERT has revolutionized natural language processing (NLP), achieving state-of-the-art performance in various tasks. Training these models, however, demands substantial computational resources, particularly large GPU cluster, making it a critical area of research for advancing AI capabilities.

Formulation of the Problem. Training LLMs on GPU cluster introduces significant challenges, including managing limited GPU memory, ensuring efficient parallelism, and minimizing communication overhead. These hurdles can lead to inefficiencies and increased costs, necessitating sophisticated methods to optimize the training process.

Analysis of Recent Research. Recent advancements have introduced several strategies for efficient LLM training on GPU cluster. Notable methods include Megatron-LM, Zero Redundancy Optimizer (ZeRO), and PyTorch Fully Sharded Data Parallel (FSDP). Each approach offers unique mechanisms to address memory limitations and parallelism, improving scalability and performance.

Highlighting Unexplored Parts of the General Problem. Despite these advancements, challenges remain in balancing memory usage, communication overhead, and computational efficiency. Further research is needed to refine these methods, especially in optimizing synchronization and reducing latency in large-scale distributed systems.

Setting Objectives. This paper aims to compare Megatron-LM, ZeRO, and PyTorch FSDP, exploring their approaches, benefits, and limitations in training LLMs on large GPU cluster. By understanding these methods, we can identify best practices and areas for further improvement.

Overview of methods. Training large language models (LLMs) on high-performance computing (HPC) servers requires specific hardware and software configurations to manage the computational demands effectively.

Megatron-LM, developed by NVIDIA, leverages model parallelism and data parallelism to train transformer-based models at scale. It efficiently splits models across multiple GPUs, using pipeline and tensor parallelism to manage large model sizes without exceeding individual GPU memory limits. This method has demonstrated near-linear scaling for models up to 1 trillion parameters on thousands of GPUs. However, it introduces complexity in synchronizing gradient updates and managing pipeline bubbles, which can reduce efficiency if not handled correctly (Shoeybi et al., 2019; Narayanan et al., 2021).

ZeRO, introduced by Microsoft, focuses on reducing memory redundancy by partitioning model states, optimizer states, and gradients across GPUs. This approach enables training extremely large models by minimizing memory usage and allowing larger batch sizes. ZeRO's stage 3 optimization allows for offloading data to CPU memory, further enhancing scalability. However, ZeRO's reliance on extensive communication can lead to high latency, particularly in large-scale clusters (Rasley et al., 2020).

PyTorch FSDP offers a flexible approach to model sharding, allowing both automatic and manual wrapping of model layers. It supports CPU offloading to improve memory efficiency and utilizes activation checkpointing to reduce memory footprint during training. FSDP achieves significant throughput improvements and enables the training of models with up to 1 trillion parameters. Despite its benefits, FSDP can face challenges related to communication overhead and the complexity of configuring sharding strategies (Lin et al., 2021).

Comparison and Analysis:

- Memory Management:
  - ZeRO – excels in reducing memory redundancy, allowing larger models to fit into available GPU memory. This is particularly beneficial when training extremely large models that would otherwise not fit in GPU memory (Rasley et al., 2020).

- FSDP also addresses memory concerns effectively with its sharding and offloading capabilities. It enables training large models by splitting model parameters and gradients across multiple GPUs and optionally offloading them to CPU memory (Lin et al., 2021).
- Megatron-LM relies on model parallelism to manage memory but can encounter pipeline bubbles that reduce efficiency. These bubbles occur during the synchronization of gradient updates across multiple GPUs (Shoeybi et al., 2019; Narayanan et al., 2021).

- Scalability:
  - Megatron-LM and FSDP have demonstrated excellent scalability, with near-linear performance improvements on large GPU clusters. Megatron-LM has been shown to scale efficiently up to 1 trillion parameters across thousands of GPUs (Shoeybi et al., 2019; Narayanan et al., 2021).
  - ZeRO also scales well but can suffer from communication overhead in extensive setups. Its stage 3 optimization helps mitigate some of this overhead by offloading data to CPU memory, allowing for larger batch sizes and improved scalability (Rasley et al., 2020).
- Communication Overhead:
  - ZeRO faces significant communication overhead due to the need for frequent synchronization of model states, optimizer states, and gradients across GPUs. This can lead to high latency, particularly in large-scale clusters (Rasley et al., 2020).
  - Megatron-LM reduces communication overhead by combining model and data parallelism but must manage synchronization across pipeline stages to minimize pipeline bubbles and idle times (Shoeybi et al., 2019; Narayanan et al., 2021).
  - FSDP introduces communication overhead when sharding model layers and during the transfer of parameters between CPU and GPU memory. However, its flexibility in configuring sharding strategies can help optimize communication patterns (Lin et al., 2021).
- Ease of Use:
  - FSDP offers a more flexible and user-friendly approach with its auto-wrapping and manual wrapping options, making it easier to integrate into existing workflows. Users can easily wrap model layers for sharding and configure offloading strategies (Lin et al., 2021).

- Megatron-LM requires more intricate configuration and synchronization. It necessitates careful management of pipeline stages and tensor parallelism to ensure efficient training (Shoeybi et al., 2019; Narayanan et al., 2021).
- ZeRO's implementation is relatively straightforward but requires careful management of communication to avoid bottlenecks. Its reliance on extensive communication for synchronization can complicate its integration into large-scale distributed systems (Rasley et al., 2020).

Conclusions. Training large language models on HPC GPU cluster is a complex task that requires efficient memory management, parallelism, and communication strategies. Megatron-LM, ZeRO, and PyTorch FSDP each offer distinct advantages and challenges. Megatron-LM excels in scalability and parallelism, ZeRO in memory optimization, and FSDP in flexibility and ease of use. Future research should focus on refining these methods to further enhance efficiency and reduce training costs, paving the way for even larger and more powerful language models.

References

1. Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, 2019. Retrieved from https://github.com/NVIDIA/Megatron-LM.

2. Rasley, J., Rajbhandari, S., Ruwase, O., & He, Y. ZeRO: Memory Optimization Towards Training A Trillion Parameter Models, 2020. Retrieved from https://arxiv.org/pdf/1910.02054.

3. PyTorch FSDP: Lin, H., Mukherjee, S., Lusted, L., Huang, M., & Deb, D. Fully Sharded Data Parallel in PyTorch, 2021. Retrieved from https://pytorch.org/docs/stable/fsdp.html.

4. Hugging Face (n.d.). Accelerate Large Model Training using PyTorch Fully Sharded Data Parallel. Retrieved from https://huggingface.co/blog/pytorch-fsdp.