Paper review: ZeRO++: Extremely Efficient Collective Communication for Giant Model Training

Yueming Yuan. yy28.

The problem the paper is trying to tackle. The paper aims to reduce the communication overhead in large-scale model training when using ZeRO, especially to address the low network bandwidth challenges. Though ZeRO is effective in scaling large language models across multiple GPUs, its communication overhead might be a bottleneck on clusters with low bandwidth or when batch sizes per GPU are small. The key problem comes from the high communication volumes in the training process, including forward and backward all-gather, and the reduce-scatter in gradient average.

The impact of the work. The problem is important because inter-node communication could be the main bottleneck in training across multiple GPU clusters. As models grow larger, efficiently using distributed computing resources becomes crucial to maintain training throughput. ZeRO++ is important because it tackles these communication inefficiencies. Especially, for relatively low-resource senarios including low bandwidth, ZeRO++ help maintain a similar throughput and accuracy comparing to the high-speed clusters. By tackling these problems, ZeRO++ help enable faster and more resource-efficient training of the massive models, which plays an important role in AI research and deployment.

The main proposed idea(s). The paper introduces three key communication volume reduction techniques.

1. Quantized weight communication for ZeRO(qwZ): In the forward pass, parameters are communicated using INT8 quantization instead of FP16, to reduce the communication volume. The blocked quantization is used to maintain accuracy.
2. Hierarchical Weight Partition for ZeRO (hpZ): This trades off GPU memory for communication by keeping a full model copy within each machine, eliminating expensive inter-node all-gather during the backward pass.
3. Quantized Gradient Communication for ZeRO (qgZ) The reduce-scatter operation is replaced by an all-to-all implementation with INT4 quantization, which further reduces communication overhead and does not significantly hurt accuracy.

Understanding of different components of the proposed technique.

1. qwZ: ZeRO partitions the model weights across all the ranks and fetches them before they are needed. ZeRO++ applies INT8 quantization to model weights during all-gather, reducing communication volume to half. To maintaining model accuracy, each weight tensor is divided into smaller chunks, and converted into INT8 by symmetric quantization, using an independent quantization scaling coefficient.
2. hpZ: For parameters, ZeRO++ keeps a secondary copy across all the devices with a machine for the backward pass. Instead of requiring all GPUs to communicate with each other across nodes, they only need to gather the model weights within a node, eliminating the inter-node communication to ZeRO in this backward all-gather.
3. qgZ: The reduce-scatter operation used in ZeRO is replaced with a hierarchical, all-to-all quantized gradient communication. The devices first apply quantization on a given tensor, then

conduct all-to-all communication among all the GPUs. Each GPU will dequantize the corresponding partial tensors and reduce on high-precision values. Since the all-to-all implementation introduces larger cross-node communication volume, they also apply the hierarchical all-to-all: first intra-node all-to-all and then followed by inter-node all-to-all.

Perceived strengths and weaknesses of the work.

Strengths:

1. Novelty: The combination of quantization and hierarchical communication strategies is novel. By addressing both model weight and gradient communication inefficiencies, ZeRO++ tackles the key bottlenecks in distributed model training.
2. Efficiency: ZeRO++ achieves significant reductions in communication volume: from 3M to 0.75M, where M is the model size. They achieve a significant throughput improvement and good scalability.
3. Usability: ZeRO++ is integrated into the existing DeepSpeed framework, making it easier for users to apply these optimizations without modifying their model code.

Weaknesses:

1. Memory: The hpZ increases memory usage significantly compared to ZeRO. This may limit the method in memory-constrained cases.
2. Accuracy: Though the methods they applied target maintaining the model quality, the introduction of quantization will still cause a reduction in model performance.
3. Evaluation: Though the experiments show strong results up to 384 GPUs, the scalability and impact of ZeRO++ at even larger scales, where inter-node communication influence is more significant, is not fully explored.

Room for Improvement.

1. The memory overhead introduced by holding full model copies within nodes could be further optimized. For example, hybrid strategies that dynamically adjust memory-communication trade-offs based on available resources could help reduce this overhead.
2. The current approach focuses on block-based quantization to maintain accuracy. It may be worth exploring other quantization techniques to achieve a better model quality.