Distributed Communication Overhead on Training GPT-350M using Megatron-LM

Jinda Jia

April 22, 2024

1 Introduction

1.1 Problem Statement

This project investigates the impact of distributed communication overhead on the performance of training the GPT-350M model using the Megatron-LM framework across various GPU configurations. These configurations include a single GPU, multiple GPUs within a single node, and multiple GPUs across multiple nodes. Specifically, I am interested in the communication overhead associated with different architectures. To explore this issue, I first conducted a bandwidth test to measure the speed of various collective communications. Subsequently, an end-to-end performance test for the distributed training of the GPT-350M model will be executed. This test will reveal the computation and communication overheads under different architectures. Based on the results, I will present the scalability trends across different settings.

1.2 Applications

The study is crucial for enhancing the efficiency of large-scale model training, which is increasingly relevant as model sizes grow.

2 Background

2.1 Megatron-LM

Megatron-LM is an open-source training framework, especially well-suited for training Large Language Models due to its excellent scalability. Training large models is challenging because a single GPU has only limited memory resources, which are insufficient to accommodate a huge model.

To address this issue, Megatron-LM incorporates a variety of distributed techniques, including Distributed Optimizer, Tensor Parallelism, Pipeline Parallelism, and Data Parallelism. These techniques are not only user-friendly but can also be seamlessly integrated to enhance the model training process.

2.2 Distributed Optimizer

In Megatron-LM, the *Distributed Optimizer* is a well-known technique used to manage the challenges associated with training large models. This is particularly important because the optimizer state can occupy a significant portion of GPU memory. For instance, when using the Adam Optimizer, the optimizer state requires $P \times 8$ bytes of memory, where P is the number of parameters.

When the *Distributed Optimizer* is enabled, gradients need to be reduced and new parameters must be gathered at each iteration. This communication is frequent and can consume a substantial amount of time, potentially becoming a major bottleneck in the training process.

3 Experiments

3.1 Environment

Big Red 200 features 640 compute nodes, each equipped with 256 GB of memory and two 64-core, 2.25 GHz, 225-watt AMD EPYC 7742 processors. Big Red 200 also includes 64 GPU-accelerated nodes, each with 256 GB of memory, a single 64-core, 2.0 GHz, 225-watt AMD EPYC 7713 processor, and four NVIDIA A100 GPUs. Big Red 200 has a theoretical peak performance (Rpeak) of nearly 7 petaFLOPS.
In this project, tests were executed on up to two nodes, each equipped with four A100 GPUs.

3.2 Nccl-test

Table 1: NCCL Benchmarking Results for All-Reduce Operations

	Inter-node (2 Nodes, 8 GPUs)		Intra-node (1 Node, 4 GPUs)	
Size (Bytes)	**Time (us)**	**BusBW (GB/s)**	**Time (us)**	**BusBW (GB/s)**
16,777,216	2990.5	9.82	205.0	122.79
33,554,432	7861.7	7.47	331.5	151.81
50,331,648	9115.2	9.66	465.3	162.25
67,108,864	14081	8.34	617.9	162.91
83,886,080	14998	9.79	743.5	169.23
100,663,296	18074	9.75	876.4	172.28
117,440,512	21282	9.66	1023.2	172.17
134,217,728	25198	9.32	1139.8	176.63
150,994,944	27169	9.73	1222.3	185.30
167,772,160	30548	9.61	1364.1	184.48
184,549,376	34752	9.29	1359.6	203.61
201,326,592	37054	9.51	1470.5	205.37

3.3 GPT-350M Pretraining

3.3.1 Parameter Setting

In this training session, we utilized the FP16 datatype to delve deeper into communication details. We set the accumulation step to 1 to facilitate a more precise analysis. Distributed Optimizer was enabled, allowing the optimizer states to be partitioned among all GPUs. The learning rate was set at 0.0003. Additionally, the data parallel size was configured to match the number of GPUs in use.

3.3.2 Communication Breakdown

To evaluate the communication overhead, we first utilized a timer to accurately measure the communication time for each iteration. Theoretically, all GPUs need to perform All-gather for weights and Reduce-scatter for gradients. Normally, the sizes of the weights and gradients should be equivalent to the size of the parameters. Since we used FP16 for training, the weights and gradients should amount to approximately $350M \times 2$ bytes, which equals 700MB.

From Figure 1, it is observed that the intra-node All-gather and Reduce-scatter operations take about 4 to 5ms each. Based on this observation, we can estimate the intra-node bandwidth as follows:

$$\text{Intra-node bandwidth} = \frac{700MB \times 3}{4 \times 5ms} = 105GB/s,$$

which is reasonable. Similarly, for inter-node communication, we calculate

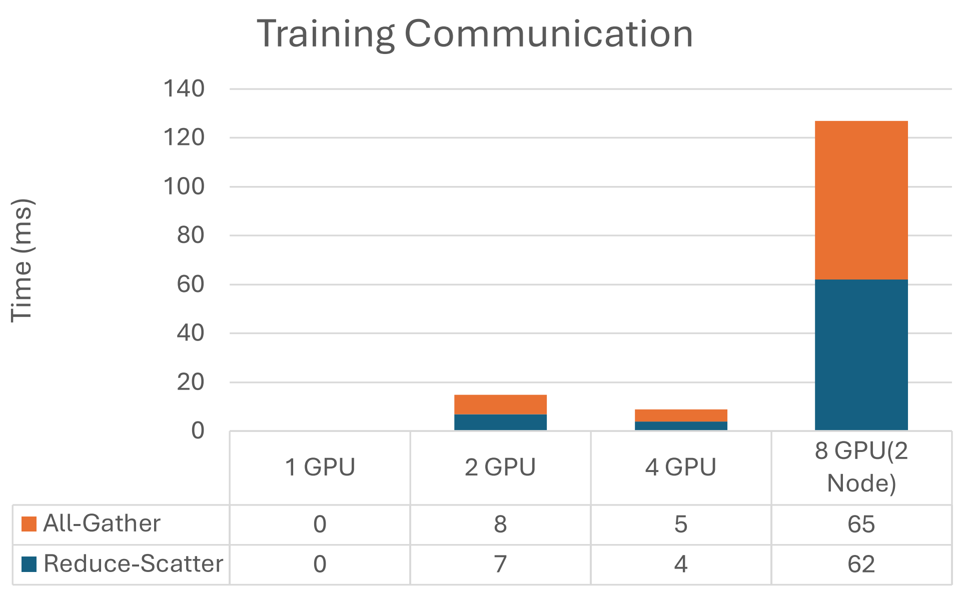
$$\text{Inter-node bandwidth} = \frac{700MB \times 7}{8 \times 62ms} = 11.29GB/s.$$

According to the results from the NCCL test, these figures are plausible.

Additionally, it is evident that when communication extends to inter-node, the communication time significantly increases. This is attributed to the relatively lower bandwidth of inter-node connections using Slingshot-10 compared to the intra-node NVLink connections.

Another observation from Figure 2 is that when the number of GPUs is fewer than four, the communication overhead is not significant compared to computation. However, when the number of GPUs increases to eight, the communication ratio increases significantly. This outcome is logical, as computation time remains stable regardless of the world size, whereas communication time is highly correlated with world size. Particularly when the number of GPUs exceeds the number available within a single node, inter-node communication constitutes a significant portion of the total overhead.

For instance, with eight GPUs, the communication ratio for training the GPT-350M model is about 15%. This might not seem substantial, and can be attributed to the relatively small model size. Since the GPT-350M is comparatively smaller, the communication overhead is less noticeable. In contrast, for larger models, such as GPT-1.3B, this communication overhead becomes considerably more significant, as illustrated in Figure 3.



	1 GPU	2 GPU	4 GPU	8 GPU(2 Node)
🟧 All-Gather	0	8	5	65
🟦 Reduce-Scatter	0	7	4	62

Figure 1: GPT-350M Communication Time Breakdown

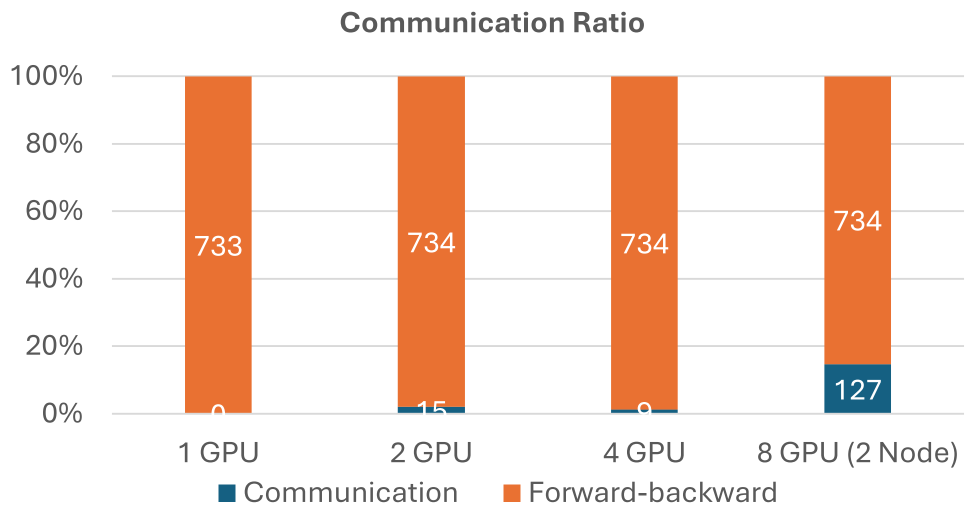


Figure 2: GPT-350M Communication Ratio

3.3.3 E2E Throughput

In this section, we discuss the end-to-end (E2E) throughput across different world sizes. To conduct this analysis, four settings were used, each with identical parameters except for the number of GPUs and the size of data parallelism. These settings range from 1 to 8 GPUs.

As depicted in Figure 4, it is observed that the E2E throughput remains relatively consistent when the number of GPUs is four or fewer. However, a significant drop in throughput is noted when the number of GPUs increases to eight. This finding is consistent with our previous tests. The slower inter-node communication inherent in larger configurations adversely

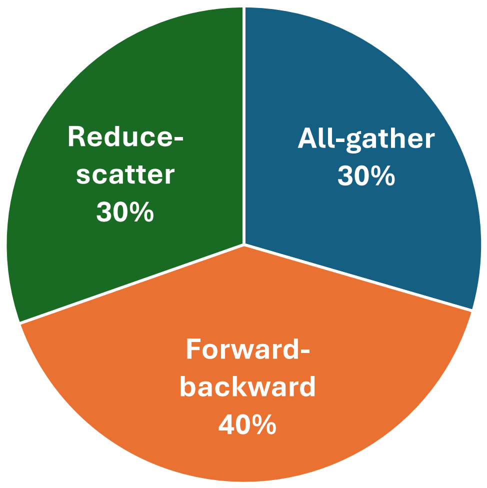


Figure 3: GPT-1.3B Communication Ratio with 8GPU

affects the E2E throughput.

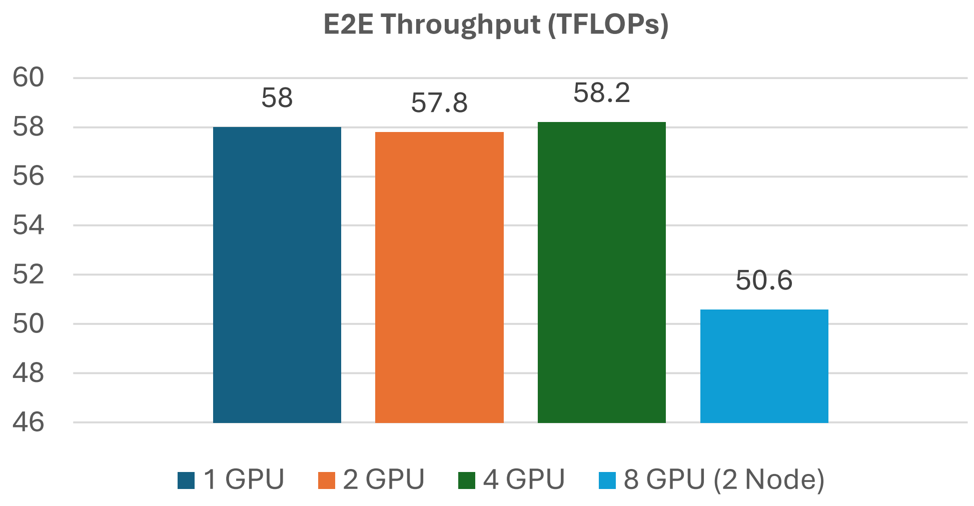


Figure 4: GPT-350M E2E Training Throughput

4 Discussion

Training a Large Language Model (LLM) on a single GPU is infeasible due to limited memory capacity and computational capability. Distributed training facilitates the partitioning of optimizer states across multiple GPUs, which enhances computational feasibility. However, this approach incurs significant communication overhead. Through this project, it has become evident that slow inter-node communication significantly hampers scalability. As the number of GPUs exceeds those available within a single node, communication overhead markedly increases, adversely affecting the end-to-end (E2E) throughput. This bottleneck is particularly pronounced as models scale up, underlining the critical impact of distributed system architecture on training efficiency.