High-Speed Data Communication With Advanced Networks in Large Language Model Training

Liuyao Dai, Hao Qi, Weicong Chen, and Xiaoyi Lu, *University of California Merced, Merced, CA, 95343, USA*

Large language models (LLMs) like Generative Pre-trained Transformer, Bidirectional Encoder Representations from Transformers, and T5 are pivotal in natural language processing. Their distributed training is influenced by high-speed interconnects. This article characterizes their training performance across various interconnects and communication protocols: TCP/IP, Internet Protocol over InfiniBand, (IPoIB), and Remote Direct Memory Access (RDMA), using data and model parallelism. RDMA-100 Gbps outperforms IPoIB-100 Gbps and TCP/IP-10 Gbps, with average gains of 2.5x and 4.8x in data parallelism, while in model parallelism, the gains were 1.1x and 1.2x. RDMA achieves the highest interconnect utilization (up to 60 Gbps), compared to IPoIB with up to 20 Gbps and TCP/IP with up to 9 Gbps. Larger models demand increased communication bandwidth, with AllReduce in data parallelism consuming up to 91% of training time, and forward receive and back-embedding AllReduce in model parallelism taking up to 90%. The larger-scale experiment confirms that communication predominates iterations. Our findings underscore the significance of communication in distributed LLM training and present opportunities for optimization.

Recently, transformer-based models, like those in Devlin et al.[1] and the GPT-4 Technical Report,[2] have excelled in natural language processing (NLP) tasks. However, their vast parameter size makes them complex and challenging. Training large language models (LLMs) demands significant computational resources due to their large number of weights, leading to the adoption of distributed training across GPUs and high-speed interconnects to expedite the process.[3] Distributed training of LLMs necessitates efficient communication among nodes and GPUs, where high-speed interconnects like InfiniBand and RoCEv2, as shown in Figure 1, are crucial for optimizing data transfer, synchronization, and overall system performance.

The sheer volume of training data and the need for distributed GPU-enabled training of LLMs further intensify the demand for high-performance interconnects, without which the communication overheads can quickly threshold the scalability and efficiency of LLM training. The past decades have witnessed the computing capability of modern high-performance computing systems scaling at more than twice the pace of interconnect bandwidth across generations. This trend raises myriad potential research problems for achieving efficient and scalable LLM training, some of which include the following:

- Will interconnects become the bottleneck for communication, and what proportion of the training process is occupied by communication for various types and configurations of LLMs?
- Are the current high-performance interconnects utilized well during different distributed training scenarios?
- What kind of quantitative performance impact will different networking technologies and protocols [such as Remote Direct Memory Access (RDMA), Internet Protocol over InfiniBand (IPoIB), and TCP/IP] have on various LLMs' training?
- What is the varying impact that different strategies or techniques, such as data parallelism and pipeline parallelism, have on networking technologies and protocols within distributed training environments?

0272-1732 © 2024 IEEE
Digital Object Identifier 10.1109/MM.2024.3360081
Date of publication 30 January 2024; date of current version 9 April 2024.

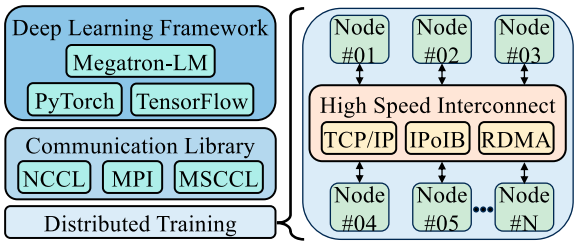

FIGURE 1. Communications and interconnects in distributed training. IPoIB: Internet Protocol over InfiniBand; RDMA: and Remote Direct Memory Access; MPI: message passing interface; MSCCL: Microsoft Collective Communication Library.

To answer these questions, this article examines the role of high-performance interconnects in distributed LLM training by employing the Megatron-LM,[3] a state-of-the-art framework that trains large-scale language models. The goal is to provide insights into the impact of various technologies and protocols on training efficiency. To achieve this goal, our methodology focuses on four pivotal dimensions: workload, interconnect/protocol, scalability, and parallelization strategies.

With our performance characterization, this article makes the following key contributions:

- We extend the breadth and depth of our experiments from our previous work[4] by applying model parallelism in experiments using up to four nodes with a total of eight A100 GPUs across all evaluated LLMs and interconnect technologies. Furthermore, in our intensified data parallelism research, we double our previous scale by involving up to four nodes with a total of 16 A100 GPUs and advanced NVLink. These expanded experiments offer deeper insights into high-speed data communication in LLMs.
- Based on our extended experiments, we evaluate key metrics, including communication latency, network bandwidth utilization, and training scalability, to assess the advantages and drawbacks of each interconnect/protocol choice for LLMs using high-speed interconnects.
- We systematically assess LLM training performance on interconnects. Our results show that high-performance protocols such as (GPUDirect) RDMA outperform others such as IPoIB and TCP/IP by 2.5× and 4.8× in data parallelism and 1.1× and 1.2× in model parallelism. Backward parameter sync can take up to 92% of the training time in data parallelism. Concurrently, forward receive and backward parameter sync in model parallelism can take up to 90%, implying that communication is a major bottleneck for distributed LLM training.
- By examining our methodology, we highlight the importance of communication in LLM training and the role of high-speed interconnects. This article offers the community detailed insight into the impact of high-performance interconnects on LLM tasks.

CHARACTERIZATION METHODOLOGY

Methodology Overview

In this section, we present an overview of the methodology used to characterize the impact of high-performance interconnects on LLMs. As mentioned earlier, our characterization focuses on four pivotal dimensions: workload, parallelization strategies, interconnect/protocol, and scalability, as shown in Figure 2(a). By systematically examining these dimensions, we gain insights into the performance and efficiency of LLM training under different parallelization strategies and different interconnect settings. The next sections provide a brief explanation of each dimension.

Workloads

In our characterization methodology, the selection of LLMs and datasets is pivotal. We focus on popular open source LLMs such as GPT-2-Medium,[5] GPT-2-Large, Bidirectional Encoder Representations from Transformers (BERT)-Large,[1] and T5-Large,[6] which span various model sizes, architectures, and application domains. OpenAI's GPT-2, the transformer-based decoder model depicted in Figure 2(b), excels in tasks like summarization and text generation. BERT, boasting up to 340 million parameters, leverages a bidirectional encoder [Figure 2(c)] to set benchmarks in NLP tasks, including text classification. T5, with as many as 11 billion parameters, utilizes a transformer architecture [Figure 2(d)] and is renowned for tasks like summarization, with its transfer learning approach being widely recognized in the NLP community. Furthermore, we incorporate a representative dataset, enwiki. Our evaluation aims to discern the impact and generalizability of high-performance interconnects across these diverse LLMs.

Parallelization Strategies

In distributed training of LLMs, two key strategies emerge: model parallelism[7] and data parallelism.[8] Model parallelism divides the model across devices, with each

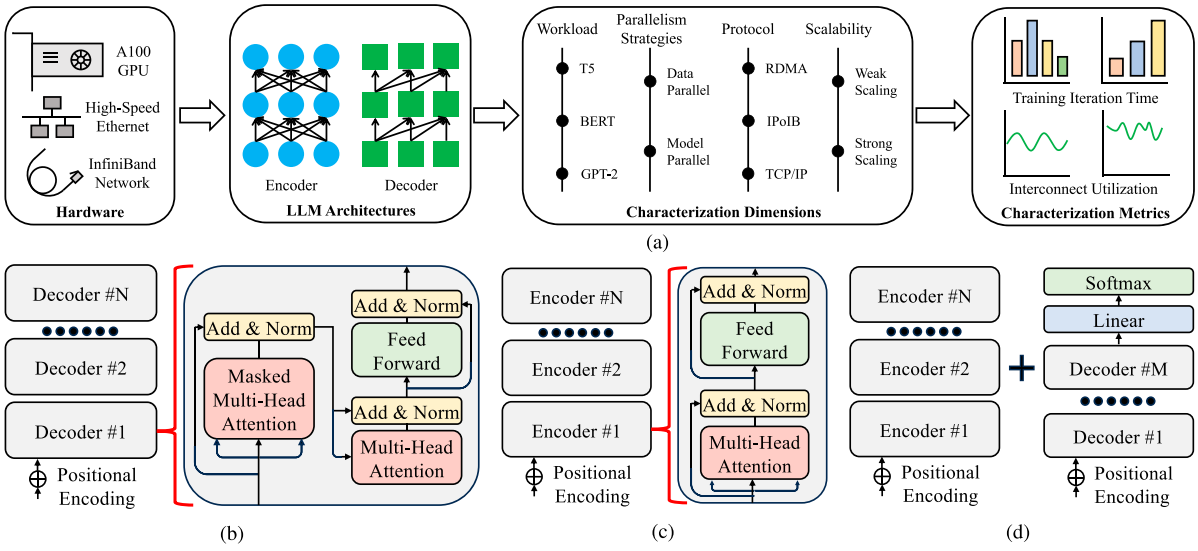

FIGURE 2. Proposed characterization methodology and transformer-based LLMs' architecture overview. (a) The characterization methodology. (b) GPT: decoder architecture. (c) BERT: encoder architecture. (d) T5: encoder and decoder architecture.

computing a model segment. Specifically, this article focuses on pipeline parallelism,[9] a type of model parallelism where the model is split into sequential stages for processing. In contrast, data parallelism distributes different data portions across devices, each running the full model on its assigned subset, expediting training on large datasets. A summary of the workload parameters for both data and model parallelism can be found in Table 1.

Interconnect/Protocol

In our methodology, we evaluate interconnect technologies such as TCP/IP,[10] IPoIB,[11] and RDMA protocol[12] (with GPUDirect). Each offers distinct performance traits and efficiencies. By analyzing their impact on LLMs, we gain insights into communication patterns and the overall performance, guiding the selection for LLM workloads.

Scalability

Scalability is crucial for evaluating LLMs in distributed settings. We assess strong and weak scaling, examining LLMs across various interconnects under data parallelism to pinpoint bottlenecks and gauge overall training efficiency.

Through our methodology's four dimensions, we evaluate how high-performance interconnects impact LLMs, offering insights into interconnect technologies and LLM performance.

Frameworks and Dataset

Framework

We use the Megatron-LM[3] framework, designed for large-scale language model training, to study the impact of high-performance interconnects on LLMs. Its support for various interconnects ensures consistent and reliable experiments, aiding in effective comparisons.

Dataset

We use the enwiki dataset (20.4 GB) from English Wikipedia in our methodology, offering diverse text for LLM training. This dataset helps evaluate LLM performance with high-performance interconnects and understand

TABLE 1. Comparison of selected LLMs in data parallelism and model parallelism.

Model	Architecture	Layers	Hidden Size	Attention Head
GPT-2 Medium	Decoder	24	1024	16
GPT-2 Large	Decoder	36*/40**	1280*/1200**	20*/24**
BERT Large	Encoder	24	1024	16
T5-Large	En/Decoder	24	1024	16

*: used only in data parallelism; **:used only in model parallelism.

the implications of parallelization in large-scale language modeling.

PERFORMANCE CHARACTERIZATION

Experimental Setup

Cluster A

The National Science Foundation (NSF)-funded Pinnacles cluster at the University of California, Merced, is equipped with eight GPU nodes. Each node has two Intel 28-Core Xeon Gold 6330 CPUs (2 GHz), 256-GB dynamic random-access memory, 2× Nvidia Tesla A100 40-GB GPUs with PCIe interface and without NVLink, and interconnected via 100-Gbps Enhanced Data Rate (EDR) InfiniBand with RDMA and 10-Gbps Ethernet. We use up to four GPU nodes in the evaluation. All the used software for four models includes Compute Unified Device Architecture (CUDA) 11.8.0, PyTorch 2.0.0, NVIDIA Collective Communications Library (NCCL) 2.14.3, Nvidia Apex 22.03, and Megatron-LM v3.0.2. This section analyzes the influence of interconnects on experiments involving data and model parallelism. We use FP16 precision training and set a global batch size=16 for strong scaling and a micro batch size=4 for weak scaling in data parallelism. Our experiments encompass configurations with one node equipped with two GPUs, two nodes with four GPUs, and four nodes with eight GPUs. The number of GPUs and the batch size have the following relationship: number of GPUs × micro batch size=-global batch size. In model parallelism, we establish different micro batch sizes for training various models.

Cluster B

Because of the limitations of the Pinnacles cluster, we conducted our experiments on another cluster designed for larger GPU scales. Each node in this cluster is equipped with four A100 GPUs. The configuration features a 600-GB/s NVLink for high-speed intranode communication, along with a 100-Gbps SlingShot connection that supports RDMA for internode communication.

Strong Scaling in Data Parallelism on Cluster A

Evaluation of Training Time

Figure 3(a)–(d) presents a comprehensive analysis of four different models across various numbers of GPUs and communication protocols/interconnects. We mainly report average numbers unless otherwise stated as models show similar performance trends. Among all the breakdowns in training iteration time, the three notable components worth discussing are forward compute time, backward compute time, and backward parameters sync time (i.e., AllReduce by NCCL).

As the GPU count rises, the models' forward computation time decreases, exhibiting 57% strong scaling efficiency. This underscores efficient multi-GPU utilization for faster forward propagation.

Similarly, increasing the GPU count reduces the backward computation time for all models, achieving 72% strong scaling efficiency. This accelerated backward propagation implies enhanced parallel processing and quicker gradient computations with more GPUs.

Although more GPUs reduce forward and backward compute times, the backward parameter synchronization via AllReduce becomes a tradeoff. As GPU count rises, the AllReduce time, essential for gradient synchronization, increases by 4.7×, largely due to added communication overhead. The efficiency of this synchronization varies with the communication protocol and interconnects, like RDMA or TCP/IP. With 8 A100 GPUs, AllReduce consumes 53%, 82%, and 92% of iteration time for RDMA, IPoIB, and TCP/IP, respectively.

> **Observation 1:** In data parallelism, the compute processes in LLM training exhibit efficient strong scaling, and the AllReduce communication process predominates in training time.

The results in Figure 3(a)–(d) reveal a tradeoff: more GPUs lead to faster computations but increased AllReduce time due to communication overhead. A scalability analysis for two, four, and eight GPUs underscores the importance of this balance, offering insights for optimal configurations.

Evaluation of Interconnect Utilization

As AllReduce for backward parameter synchronization consumes significant training time, influenced by protocols and interconnects, we proceed to analyze achieved interconnect utilization using system counters. Illustrated here are interconnect utilization figures for eight GPUs, showcased in Figure 3(e)–(h). This choice highlights the experiment with the highest utilization, offering a clearer performance picture of the interconnect. The assessment evaluates receiver (Rx) and transmitter (Tx) speeds while training four models using RDMA, IPoIB, and TCP/IP. We present only the Rx results in Figures 3 and 4 as the Tx and Rx speeds were almost identical in our data parallelism experiment. Due to their similarity, IPoIB's results partially represent TCP/IP's performance on high-speed interconnects.

For the two protocols using 100-Gbps InfiniBand, RDMA shows the highest speeds, ranging from 30 to

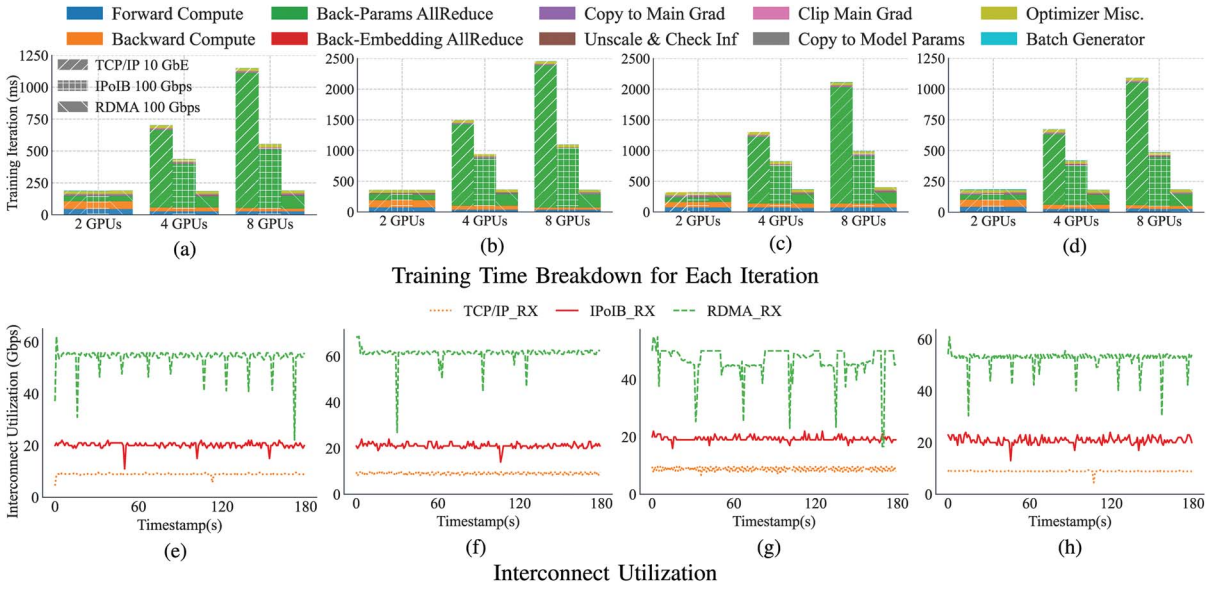

FIGURE 3. Training time breakdown for each iteration. (a) GPT-2-Medium. (b) GPT-2-Large. (c) T5-Large. (d) BERT-Large. Interconnect utilization using data parallelism under *strong scaling* on cluster A. (e) GPT-2-Medium. (f) GPT-2-Large. (g) T5-Large. (h) BERT-Large. params: parameters; misc.: miscellaneous; Rx: receiver.

60 Gbps for both the Rx and Tx. This indicates that RDMA can efficiently utilize the available bandwidth. On the other hand, IPoIB exhibits slightly lower speeds, with the Rx and Tx ranging from 17 to 20 Gbps. In contrast, TCP/IP shows the lowest speeds among the interconnect/protocol options, with the Rx and Tx speeds ranging from 8 to 9 Gbps.

It is worth noting that occasional drops in Rx and Tx speeds are observed, resulting from checkpointing and validation processes during model training.

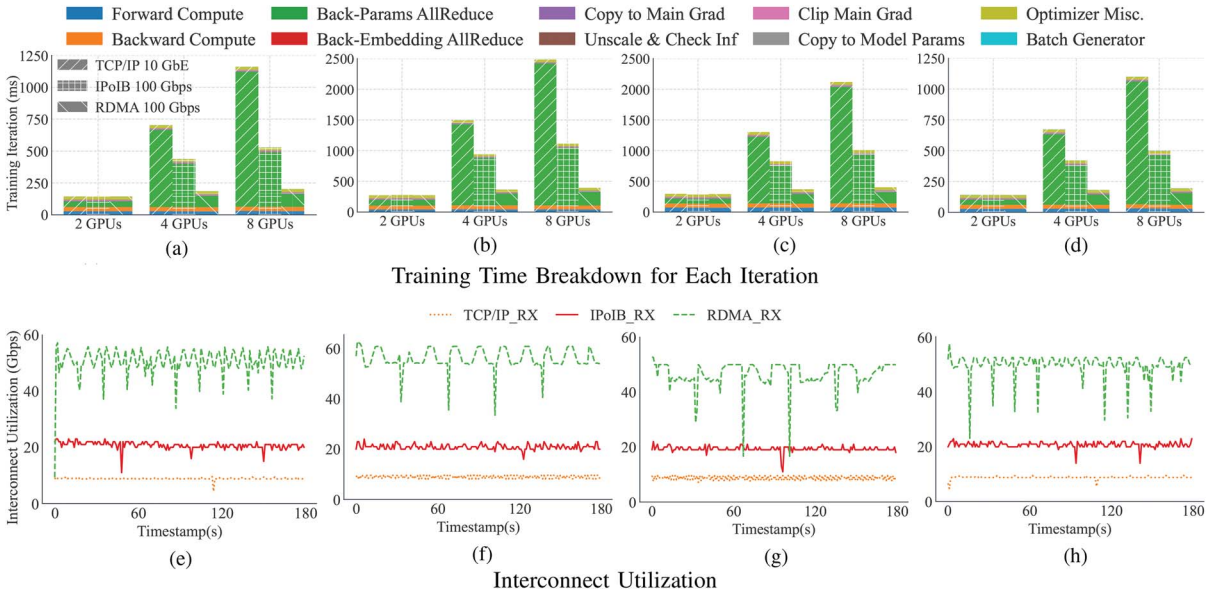

FIGURE 4. Training time breakdown for each iteration. (a) GPT-2-Medium. (b) GPT-2-Large. (c) T5-Large. (d) BERT-Large. Interconnect utilization using data parallelism under *weak scaling* on cluster A. (e) GPT-2-Medium. (f) GPT-2-Large. (g) T5-Large. (h) BERT-Large. params: parameters; misc.: miscellaneous; Rx: receiver; Grad: gradients; Inf: infinite.

Checkpointing involves saving intermediate states, and validation assesses model performance on a separate dataset. These operations intermittently affect communication and processing speeds, leading to periodic Rx and Tx speed drops.

We observe GPT-2-Large consistently achieves higher Rx and Tx speeds (30.47 Gbps) within the models tested than other models, like GPT-2-Medium (19.93 Gbps), BERT-Large (26.48 Gbps), and T5-Large (24.19 Gbps). This is because it is relatively large among all the models in our experiments and thus necessitates more data communication. T5-Large suffers more severe periodic utilization drops, albeit in comparable size, which hinders its overall interconnect utilization. This suggests that GPT-2-Large is more efficient in utilizing the interconnect bandwidth and demonstrates better scaling performance in the given strong scaling scenario.

Observation 2: In data parallelism, the utilization of interconnects for training LLMs follows the order of RDMA> IPoIB> TCP/IP. Larger LLMs demonstrate increased interconnect utilization.

To summarize, the overall interconnect utilization is less than desirable, given that we have 100-Gbps InfiniBand and 10-Gbps Ethernet. Therefore, future analyses may explore the interplay among the models, GPU configurations, and interconnect/protocol options to gain a deeper insight into their performance characteristics and identify strategies for maximizing interconnect utilization.

Weak Scaling in Data Parallelism on Cluster A

Evaluation of Training Time

In this section, we evaluate the weak scaling of those models across different GPU configurations, specifically, two, four, and eight GPUs. We report average numbers due to similar strong scaling trends unless stated otherwise. From Figure 4(a)–(d), we see that the training iteration time depends mainly on the forward compute time, backward compute time, and backward parameter synchronization time.

Under varying numbers of GPUs, all the models demonstrated excellent scalability regarding forward compute time and backward compute time, with 97% and 99%, respectively. As the GPU count increased from two to four to eight, these models continued to scale well, with the forward and backward compute times remaining relatively constant. This indicates that all the models can handle larger problem sizes without significantly increasing compute time.

However, it is worth noting that the models showcased varying weak scaling characteristics regarding AllReduce operation. Across all models, the overall trend remains the same: AllReduce time is heavily influenced by the protocols/interconnect. Specifically, RDMA outperforms IPoIB and TCP/IP by 4.1× and 8.7×, respectively, leading to 2.5× and 4.8× faster training iterations, respectively.

Among the models, GPT-2-Large and T5-Large demonstrate longer AllReduce time across all GPU configurations, indicating potential communication challenges during the parameter synchronization step. On the other hand, GPT-2-Medium and BERT-Large showcase a relatively shorter AllReduce time. Using eight A100 GPUs, AllReduce time takes up to 51%, 81%, and 91% of iteration time for RDMA, IPoIB, and TCP/IP, respectively.

Observation 3: In the weak scaling of data parallelism, compute times remain consistent as the number of nodes increases from one node (to GPUs) to four nodes (eight GPUs). Additionally, AllReduce communication continues to dominate the training time of LLMs.

In conclusion, although AllReduce time relies heavily on interconnect types, all the models' forward and backward compute times scale well. These observations highlight the importance of considering the interconnect's characteristics when assessing the scalability of distributed training for LLMs while showcasing the models' ability to scale efficiently in forward and backward computations.

Evaluation of Interconnect Utilization

Analogously, we showcase the interconnect utilization under a weak scaling experiment with eight GPUs in Figure 4(e)–(h), providing insights into the performance characteristics of different models and interconnect/protocol options. With our analysis, several pivotal findings emerge.

The weak scaling trend initially follows a similar pattern to the strong scaling experiment across all models and interconnect/protocol options. As the number of GPUs increases from four to eight, the interconnect utilization improves: in an eight-GPU setup, it is 1.1×, 1.4×, and 1.7× higher than in a four-GPU setup for TCP/IP, IPoIB, and RDMA, respectively. This suggests that incorporating additional GPUs may necessitate

the utilization of additional available interconnect resources.

Among the tested models, GPT-2-Large consistently exhibits the highest utilization (28.6 Gbps), reaffirming its demand to utilize the interconnect effectively. This can be attributed to its larger model size and computational workloads, rendering more use of the available computational resources and interconnect bandwidth.

Regarding interconnect/protocol options, RDMA consistently outperforms IPoIB and TCP/IP. RDMA, with its efficient direct memory-access capabilities, achieves the highest Rx and Tx speeds (38–56 Gbps). IPoIB follows with slightly lower speeds (17–20 Gbps), while TCP/IP exhibits the lowest speeds (8–9 Gbps). This hierarchy emphasizes again the importance of choosing the appropriate interconnect/protocol option for achieving optimal interconnect utilization.

Observation 4: The interconnect utilization during weak scaling continues to follow the order of RDMA> IPoIB> TCP/IP.

Additionally, periodic drops in Rx and Tx speeds are observed in the data, analogous to the strong scaling experiment. These drops occur due to the checkpointing and loss validation processes necessary to maintain the training process's integrity and accuracy. A heavier workload of eight GPUs incurs more pronounced Rx and Tx speed drops. Despite being temporary, these drops reflect the tradeoff between performance and ensuring the quality of the training process.

In conclusion, the weak scaling experiment with eight GPUs reinforces the importance of interconnect utilization in scaling scenarios. The result highlights GPT-2-Large's higher interconnect utilization, the advantage of RDMA over IPoIB and TCP/IP, and the presence of periodic drops in receive and send speeds. These findings contribute to a deeper understanding of the performance characteristics of models and interconnect/protocol options in the context of weak scaling. Further analysis and experimentation can build upon these insights to optimize interconnect utilization in different scaling scenarios.

Model Parallelism on Cluster A

This section evaluates various models across different GPU configurations in model parallelism, including two, four, and eight GPUs. As observed in Figure 5(a)–(d), the training iteration time depends primarily on the forward compute time, backward compute time, forward receive time, and backward-embedding synchronization time. The forward receive time denotes the duration to receive input data for the forward pass through the pipeline stages. The *backward-embedding synchronization time* refers to the time needed for AllReduce relative to embedding layers during backward passes across the pipeline stages. The global batch sizes for BERT-Large, GPT-2-Medium, and T5-Large have been configured to 32, 64, and 128 for systems using two, four, and eight GPUs, respectively. In the case of GPT-2-Large, the global batch sizes are specifically set to 16, 32, and 64 for configurations with two, four, and 8 GPUs, respectively.

With an increase in the number of GPUs, all the models experience an increase in both forward and backward compute times.

A consistent trend is observed across all the models: both the forward receive time and the AllReduce time are influenced by the protocols or interconnect. Specifically, RDMA outperforms TCP/IP and IPoIB by up to 1.4× and 1.2× for the forward receive time, respectively, and by up to 1.4× and 1.2× for the AllReduce time, respectively.

In model parallelism, network communications are a key factor in LLM training. Among the models, the forward receive times and AllReduce times for

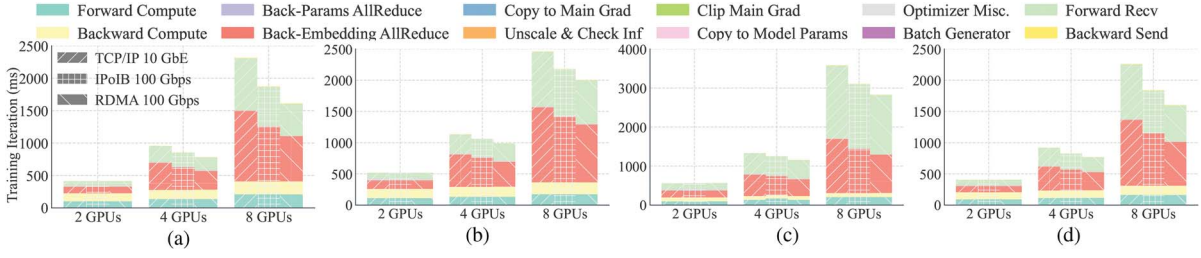

FIGURE 5. Training Time Breakdown for Each Iteration in Model Parallelism on Cluster A. (a) GPT-2-Medium. (b) GPT-2-Large. (c) T5-Large. (d) BERT-Large. params: parameters; misc.: miscellaneous; recv: receive.

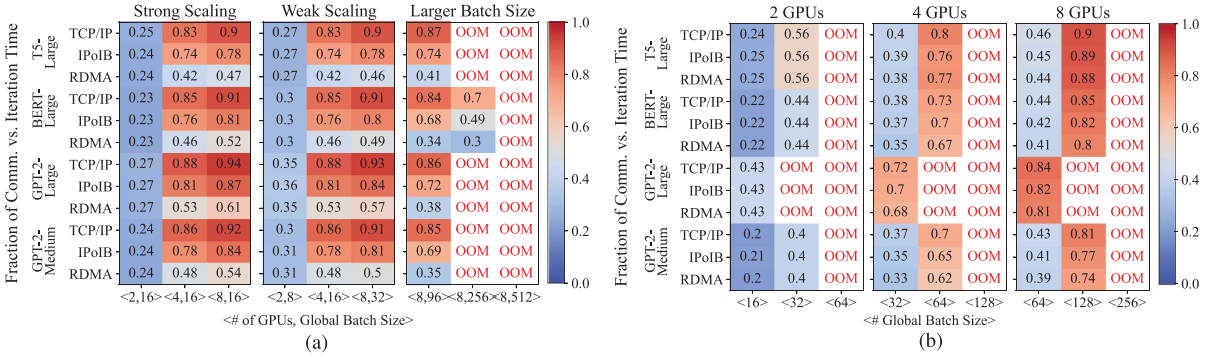

FIGURE 6. Fraction of communication versus iteration time with larger batch sizes in different parallelisms on cluster A. (a) Data parallelism and (b) model parallelism. OOM: out of memory.

T5-Large represent the largest fraction of the total time, which may indicate communication challenges during the parameter synchronization step.

Observation 5: During LLM training with model parallelism, as the number of nodes increases from one node (two GPUs) to four nodes (eight GPUs), there is a corresponding increase in compute time. Despite this, communication time continues to dominate the training duration, and the training speed hierarchy remains as follows: RDMA> IPoIB> TCP/IP.

In conclusion, the execution duration for forward receive and AllReduce vary across different interconnect types, and the computation time in model parallelism may present scalability challenges. These observations underscore the importance of selecting the most efficient interconnect type to circumvent scalability issues in model parallelism.

LLMs WITH MORE PARAMETERS TEND TO SHOW HIGHER INTERCONNECT UTILIZATION REQUIREMENTS IN DATA PARALLELISM, BUT THERE IS STILL ROOM FOR THE OVERALL INTERCONNECT UTILIZATION.

Data and Model Parallelism Summary on Cluster A

As shown in Figure 6(a), data parallelism is examined by further evaluating these models, incrementally increasing the batch sizes until an out-of-memory (OOM) error occurs. This analysis shows a consistent trend where communication consumes a significant portion of the iteration time. Notably, this figure demonstrates that even though increasing the batch sizes can result in a reduced proportion of communication time in the overall iteration time (amortized by the prolonged computation time), the communication time can still occupy at least 34% of iteration time, except for BERT-Large, which allows for much larger batch sizes.

In Figure 6(b), we showcase model parallelism and strong scaling by increasing the number of GPUs while keeping a consistent global batch size, and expanding batch sizes until an OOM error arises. We observe that communication proportion rises with batch-size growth, contrasting with data parallelism. With the same batch sizes, more GPUs lead to reduced communication fractions. However, the fraction of communication still constitutes at least 39% of the iteration time for a global batch size of 64, and 74% of the iteration time for a micro global size of 128. This exception applies to GPT-2-Large, which does not permit a global batch size of 128 when utilizing eight GPUs.

This observation highlights a lower bound of the communication time proportion because it investigates until the maximum batch size a model can train with at the given scale.

Data Parallelism in Larger-Scale GPU Configurations on Cluster B

This section expands our evaluation to larger GPU configurations by testing the GPT-2-Large model across four nodes with 16 A100 GPUs. This model is one of the most substantial open source models available to the research community. Due to the constraints of the Pinnacles cluster, we conducted our experiments on cluster B, utilizing the RDMA protocol.

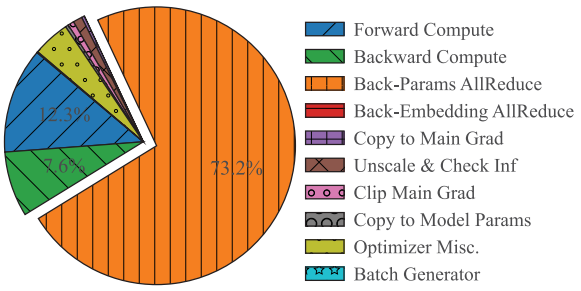

FIGURE 7. Training time breakdown for 16 A100 GPUs in data parallelism on cluster B. params: parameters; misc.: miscellaneous.

Addressing the aspects of data parallelism, our analysis indicates that the upper limit for the global batch size of GPT-2-Large on an eight-GPU setup is 96. There is a direct proportionality between the number of GPUs and the batch size, governed by the following formula: number of GPUs × micro batch size=global batch size. Based on this relationship, we set the global batch size to 192 for our current assessment. Our findings highlight that AllReduce operations account for 73% of the iteration time, as shown in Figure 7, which aligns with the findings from our experiments on eight GPUs with a global batch size of 96, thereby emphasizing the consistent impact of communication processes at varying scales of GPU utilization.

Observation 6: In data parallelism with 16 A100 GPUs, the communication overhead continues to predominate over iteration time as the scale of GPU configurations increases.

CONCLUSION AND FUTURE WORK

This article contributes to understanding the performance characteristics of LLMs over high-speed interconnects. Our exploration and summarization of communication's role in distributed LLM training yield valuable insights. These insights inform efficient system design to support the growing demand for LLM applications.

The evaluation yields noteworthy insights, emphasizing trends' similarity and disparity in data parallelism and model parallelism experiments. This underscores interconnects' and protocols' impact on distributed training. Notable takeaways include the following:

- Forward and backward computation times exhibit favorable scaling in the context of data parallelism, but less so in model parallelism. Nonetheless, combining data parallelism and model parallelism in distributed LLM training introduces scalability challenges related to communication.
- Faster interconnects/protocols can significantly reduce the distributed training time for LLMs. Notably, GPUDirect RDMA surpasses IPoIB and TCP/IP by 2.5× and 4.8×, on average, in data parallelism for training performance, respectively. In model parallelism, GPUDirect RDMA outperforms IPoIB and TCP/IP by 1.1× and 1.2×, on average, in training performance, respectively.
- LLMs with more parameters tend to show higher interconnect utilization requirements in data parallelism, but there is still room for the overall interconnect utilization to be improved, even under heavy-LLM workloads.
- As the number of GPUs scales from two to 16, the communication time, particularly the time spent in AllReduce operations, continues to predominate the total iteration time.

AS THE NUMBER OF GPUs SCALES FROM TWO TO 16, THE COMMUNICATION TIME, PARTICULARLY THE TIME SPENT IN ALLREDUCE OPERATIONS, CONTINUES TO PREDOMINATE THE TOTAL ITERATION TIME.

Some of the future work may include investigating the interconnect utilization in model parallelism, exploring distributed training behavior for even larger models at larger scales, and developing techniques to further optimize interconnect utilization.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable insights, thoughtful comments, and constructive feedback. Their expertise and thorough evaluation significantly contributed to the improvement of this work. This work was supported in part by NSF research Grant Office of Advanced Cyberinfrastructure (OAC) 2321123 and Grant OAC 2340982, an Amazon research award, and U.S. Department of Energy research Grant DE-SC0024207. Part of this research was conducted using Pinnacles (NSF MRI, 2019144) at the Cyberinfrastructure and Research Technologies at the University of California, Merced. Liuyao Dai and Hao Qi are co-first authors.

REFERENCES

1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
2. "GPT-4." OpenAI. [Online]. Available: https://openai.com/research/gpt-4
3. D. Narayanan et al., "Efficient large-scale language model training on GPU clusters using megatron-LM," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, New York, NY, USA: ACM, 2021, pp. 1–15, doi: 10.1145/3458817.3476209.
4. H. Qi, L. Dai, W. Chen, Z. Jia, and X. Lu, "Performance characterization of large language models on high-speed interconnects," in *Proc. IEEE Symp. High-Perform. Interconnects (HOTI)*, 2023, pp. 53–60, doi: 10.1109/HOTI59126.2023.00022.
5. A. Radford et al., "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
6. C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 5485–5551, 2024.
7. J. Dean et al., "Large scale distributed deep networks," *in Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1223–1231.
8. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 84–90.
9. Y. Huang et al., *GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism*. Red Hook, NY, USA: Curran Associates Inc., 2019.
10. V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *IEEE Trans. Commun.*, vol. 22, no. 5, pp. 637–648, May 1974, doi: 10.1109/TCOM.1974.1092259.
11. V. Kashyap, "IP over InfiniBand (IPoIB) architecture," RFC Editor, Internet Engineering Task Force (IETF), Tech. Rep., RFC 4392, Apr. 2006.
12. R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia, "A remote direct memory access protocol specification," RFC Editor, RFC 5040, Tech. Rep., 2007.

LIUYAO DAI is a Ph.D. student in the Parallel and Distributed Systems Laboratory of the Department of Computer Science and Engineering at the University of California, Merced, Merced, CA, 95343, USA. His research interests include GPU-aware collective communication libraries. Dai received his master's degree from Southern University of Science and Technology. Contact him at ldai8@ucmerced.edu.

HAO QI is a Ph.D. student in the Parallel and Distributed Systems Laboratory of the Department of Computer Science and Engineering at the University of California, Merced, Merced, CA, 95343, USA. His research interests include high-performance computing, parallel computing, and systems for machine learning. Qi received his master's degree from The Ohio State University. Contact him at hqi6@ucmerced.edu.

WEICONG CHEN is a postdoctoral scholar in the Parallel and Distributed Systems Laboratory of the Department of Computer Science and Engineering at the University of California, Merced, Merced, CA, 95343, USA. His research interests include high-performance computing, big data, and Bayesian methods. Chen received his Ph.D. degree from Case Western Reserve University. Contact him at wchen97@ucmerced.edu.

XIAOYI LU is an assistant professor in the Department of Computer Science and Engineering at the University of California, Merced, Merced, CA, 95343, USA, where he leads the Parallel and Distributed Systems Laboratory. His research interests include parallel and distributed computing, high-performance interconnects, and deep learning system software. He is a Member of IEEE and Association for Computing Machinery. He is the corresponding author of this article. Contact him at xiaoyi.lu@ucmerced.edu.