Performance of Distributed Deep Learning Workloads on a Composable Cyberinfrastructure

Zhenhua He*
HPRC, Texas A&M University,
College Station TX
happidence1@tamu.edu

Aditi Saluja
HPRC, Texas A&M University,
College Station TX
saluja.aditi5@tamu.edu

Lisa M. Perez
HPRC, Texas A&M University,
College Station TX
perez@tamu.edu

Francis Dang
HPRC, Texas A&M University,
College Station TX
francis@tamu.edu

Dhruva K. Chakravorty
HPRC, Texas A&M University,
College Station TX
chakravorty@tamu.edu

Honggao Liu
HPRC, Texas A&M University,
College Station TX
honggao@tamu.edu

ABSTRACT

The next generation of computing systems are likely to rely on disaggregated resources that can be dynamically reconfigured and customized for researchers to support scientific and engineering workflows that require different cyberinfrastructure (CI) technologies. These resources would include memory, accelerators, co-processors among other technologies. This would represent a significant shift in High Performance Computing (HPC) from the now typical model of clusters that have these resources permanently connected to a single server. While composing hardware frameworks with disaggregated resources holds promise, we need to understand how to situate workflows on these resources and evaluate the impact of this approach on workflow performance against "traditional" clusters. Toward developing this knowledge framework, we study the applicability and performance of deep learning workloads on GPU-enabled composable and traditional HPC computing platforms. Results from tests performed using the Horovod framework with TensorFlow and PyTorch models on these HPC environments are presented here.

CCS CONCEPTS

• **Composable**; • **High Performance Computing**; • **Distributed Deep Learning**; • **AI/ML**; • **Performance Benchmarks**;

KEYWORDS

FASTER (Fostering Accelerated Sciences Transformation Education and Research), Grace, ResNet50, BERT-Large, Accelerators, GPU (Graphics Processing Unit), A100, T4

ACM Reference Format:
Zhenhua He, Aditi Saluja, Lisa M. Perez, Francis Dang, Dhruva K. Chakravorty, and Honggao Liu. 2023. Performance of Distributed Deep Learning Workloads on a Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing (PEARC '23), July 23–27, 2023, Portland, OR, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3569951.3593601

*High Performance Research Computing.

PEARC '23, July 23–27, 2023, Portland, OR, USA

ACM ISBN 978-1-4503-9985-2/23/07…$15.00
https://doi.org/10.1145/3569951.3593601

1 INTRODUCTION

Advanced computing increasingly plays a determining role in guiding innovation and discovery in research. Next generation processor and accelerator technologies are reshaping the way Science and Engineering (S&E) disciplines are applying numerical simulation techniques, and artificial intelligence and machine learning (AI/ML) frameworks in discovery pathways. We are witnessing a convergence of AI/ML and traditional high-performance computing (HPC) workloads coming together in several positive ways. Today, we have arrived at a junction where traditional computing approaches can combine with AI/ML frameworks to glean novel insights from data [1]. We are seeing the emergence of AI systems that perform part of the software development process for complex systems. The continuing demand for HPC platforms that seamlessly support AI/ML workloads have required a computing, analysis, and data-storage solution that simultaneously satisfies the needs of AI, IoT (internet of things), edge-processing, instrument, and sensor data analytics, along with traditional HPC workloads. Computational science, the "third pillar" of research and scientific investigation, today drives theory and experimentation [2]. Indeed, as we boldly step into the "Fourth Paradigm", the Data-Intensive Scientific Discovery, analytics and data offer unparalleled opportunities for researchers to make path-breaking discoveries [3-6]. As part of this, future computing systems are moving towards a model of disaggregation and composition, where resources are dynamically composed, instead of relying on traditional clusters [7, 8]. To support this trend, technology companies such as Liqid [9] and GigaIO [10] have developed Matrix software and FabreX technology, respectively, to enable the composability of data center resources. There are different types of composability, which can include physical forms such as the National Science Foundation (NSF)-funded Accelerating Computing for Emerging Sciences (ACES) [11] and National Research Platform (NRP) [12] computing systems, as well as software-defined forms such as NSF-funded Anvil system with Kubernetes. However, despite the potential benefits of composable computing systems, there is a lack of performance benchmarks to

fully evaluate their performance. Benchmarking is crucial for evaluating and optimizing the performance of a computing system or cyberinfrastructure [13]. It has been indicated that Composable Disaggregated Infrastructure (CDI) can be designed without sacrificing performance compared to traditional clusters [14].

The NSF-funded (Fostering Accelerated Sciences Transformation Education and Research) FASTER composable computing cluster at Texas A&M University provides a unique platform to evaluate the performance of distributed deep learning workloads in such an environment [15]. Here we visit different scenarios that researchers may use to run artificial intelligence and machine learning frameworks on GPUs. For this, we perform a variety of performance calculations on popular models and frameworks. Specifically, we analyze the scaling behavior of ResNet50 [16] and BERT-Large [17] on NVIDIA GPUs (A100 and T4) [18, 19] with Intel ice lake processors in composable environments to that in traditional heterogeneous CPU-GPU clusters with networking enabled using InfiniBand technologies.

1.1 Traditional vs. Composable High Performance Computing Server Layout

In a traditional HPC layout, resources such as CPU, memory, devices, and storage are dedicated to the nodes in a static manner. Traditional HPC layouts are static, as they rely on fixed, dedicated resources that are difficult to reconfigure, which can lead to overprovisioning of resources and a lack of flexibility to adapt to different workloads. Traditional HPC layouts are observed on NSF-funded clusters such as NSF Expanse [20], Anvil [21] and Delta [22] machines. Figure 1 illustrates the difference between traditional and composable HPC (High-Performance Computing) approaches to hardware usage in advanced research computing. In a composable HPC layout, nodes can be composed with the needed resources (accelerators, memory, storage) to meet the expectations of the workloads. The composable HPC layout offers some advantages over traditional HPC layouts. Computing resources can be dynamically provisioned and reallocated from a common resource pool, allowing data centers to optimize their resources and adapt to various types of workloads more easily. For scalability, composable HPC can be scaled up or down to meet the demands of different workloads as well. How these advantages help real scientific workflows employ such dynamically composing disaggregated infrastructure (CDI) needs to be evaluated.

Grace [23] and FASTER are two supercomputing clusters hosted at Texas A&M High Performance Research Computing (HPRC). The Grace supercomputing cluster is a traditional HPC platform at Texas A&M HPRC that is similar to a number of NSF-funded clusters allocated via ACCESS. It is a 925-node (44,656 total cores) Linux cluster with Intel Cascade Lake processors and NVIDIA GPU nodes (A100, RTX6000, and T4). In the Grace cluster, there are GPU nodes with either two A100s, two RTX6000, or 4 T4 GPUs per node with either 1 or 2 GPUs attached per Intel Cascade Lake Xeon socket via PCIe 3.0 x16. The FASTER supercomputing cluster is a composable HPC platform with 180-nodes featuring Intel Ice Lake processors, Mellanox HDR100 InfiniBand, and NVIDIA A100, A10, A30, A40 and T4 GPUs. Each node is equipped with CPUs (64 cores), 256 GB RAM, and 3.8 TB NVMe local storage. The Liqid PCIe Gen4-based Matrix software and fabric technology is used to manage composability on the FASTER cluster. The fabric technology provides a shared, centralized resource pool that enables resources to be dynamically allocated and reallocated as needed, allowing administrators to provision, manage, and allocate resources to meet changing workload demands.

On a traditional cluster, each GPU is connected to a server socket by a single uninterrupted PCIe connection. In the Liqid fabric, GPUs are connected by a route involving multiple PCIe connections, and some of those connections are shared. i.e., if the nodes want to communicate with two different GPUs, the routes might overlap on one of the PCIe connections, which may cause a bottleneck. Within a server rack on FASTER, GPUs are physically housed in enclosures separate from the rack mount servers. The GPU enclosures and servers are interconnected by 2-3 PCIe 4.0 switches. Within each switch, there are three chips with six PCIe 4.0 x16 connections per chip, where each connection supports 64 GB/s bidirectional bandwidth. Two connections are used to interconnect the three chips within a switch in an "east/west" fashion. The other four connections are available for external connectivity to host nodes, GPUs, and other switches. The net result of this is a web-like topology wherein all of the devices in the rack are connected, but some more distant devices are connected through longer routes passing through multiple PCIe connections. Different numbers of GPUs on a single node impact code performance on CDIs, introducing a new variable to scaling studies. With it being possible to compose nodes with different number of GPUs, it is possible to leverage both PCIe and InfiniBand lanes of connections between nodes. One can now achieve the total number of accelerators (GPUs) for a task using different combinations of composed nodes. As a result, on CDI clusters like FASTER, we not only need to perform common weak and strong scaling studies to understand how a code scales on GPUs, but also have to visit the configuration of the nodes.

1.2 Distributed Deep Learning Framework – Horovod

Horovod is a popular framework-agnostic distributed deep learning training framework for TensorFlow, PyTorch, Apache MXNet, among others [24]. It has user-friendly utilities and requires minimal code changes. These advantages make it a flexible choice for researchers who want to use their existing codebase. It is designed to facilitate distributed deep learning. Horovod supports model parallelism and data parallelism, letting researchers train large deep learning models on multiple GPUs. In this study, we run TensorFlow and PyTorch ResNet50 models in the Horovod framework. In addition to Horovod, there are also other frameworks for distributed deep learning training such as TensorFlow Distributed, PyTorch Distributed, Apache MXNet, etc. While Horovod shares many similarities with those frameworks for distributed deep learning training, it benefits from using "ring-allreduce" to efficiently synchronize gradient updates across multiple workers. This allows it to scale up to hundreds or even thousands of workers without sacrificing performance.

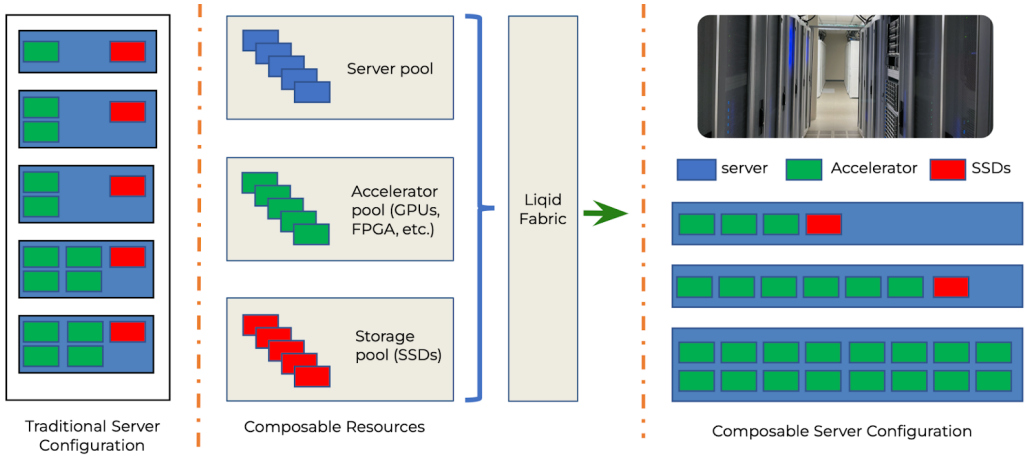

Figure 1: Traditional and Composable HPC layout

2 METHODS

2.1 Benchmarking Models

ResNet50 and BERT-Large are two well-established models that can be trained on large datasets. They have been widely used for various research applications in the fields of computer vision and natural language processing (NLP). They can be used as base models for transfer learning, where the pre-trained models are fine-tuned on a smaller labeled dataset specific to a research task. ResNet50 is a convolutional neural network that can be trained on the ImageNet dataset, which contains over 14 million images and 1000 classes. It is known for its ability to perform image classification tasks, and its deep architecture with 50 layers. BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based neural network that can be trained on a large corpus of text data. It is designed to perform a wide range of natural language processing (NLP) tasks that include answering questions and sentiment-analysis. The transformer layer is the fundamental building block of the BERT architecture, which allows the model to process and understand the relationships between words in a sentence. BERT has two main versions: BERT-Base and BERT-Large. BERT-Base has 12 transformer layers, while BERT-Large has 24 transformer layers. There are other variants of BERT such as ALBERT (a Lite BERT) [25], DistilBERT (Distillation BERT) [26], and Tiny BERT [27]. These variants have fewer hyperparameters and are more suited for systems with lesser compute capabilities. The additional layers in BERT-Large make it a more powerful and capable model, but also require more computational resources and training time. Training BERT models is computationally intensive and is typically done on GPUs. While training these models with PyTorch Distributed Data Parallel (DDP), each GPU processes a different subset of the training data and computes the gradients with respect to its subset of the weights. These gradients are then communicated to a "master" GPU that aggregates the gradients and updates the weights. Training these models pushes the limits of a GPU's performance, making them useful as aides to benchmarking the capabilities of different GPUs. For our experiments, we have used the BERT-Large model. The TensorFlow ResNet50 model, PyTorch ResNet50 model, and PyTorch BERT-Large model code can be found in [28], [29], [30], respectively.

2.2 Benchmarking Environments Setup

The benchmarking environment for the Horovod TensorFlow ResNet50 model was prepared using Horovod - v0.22.1, TensorFlow - v2.6.0, CUDA - v11.3.1, GCC - v10.3.0, and OpenMPI -v4.1.1. on the FASTER and Grace clusters. The testing environment for Horovod PyTorch ResNet50 model was prepared using Horovod - v0.22.0, PyTorch - v1.8.1, GCC - v10.2.0, CUDA - v11.1.1, and OpenMPI - v4.0.5 on the FASTER and Grace clusters as well. The testing environment for PyTorch BERT-Large model on the nodes was set up using the Singularity container runtime engine. The nvidia/PyTorch 21.10-py3 container image was obtained from the NVIDIA container registry [31].

3 RESULTS

3.1 Horovod TensorFlow and PyTorch ResNet50 Model

To evaluate the performance of deep learning workloads on composable and traditional HPC cyberinfrastructures, we ran Horovod TensorFlow ResNet50 model on FASTER and Grace supercomputing clusters. The heterogeneous CPU-GPU Grace cluster has 2 GPUs (NVIDIA A100 GPUs 40GB) on each node, and nodes are connected using infiniband. Nodes on the FASTER cluster were composed with e A100 GPUs. The HPL Linpack test using Nvidia's HPC benchmark container on NGC [32] for GPU computation was executed to ensure that the node was composed correctly.

Here, we examine the performance in the single-node and multi-node settings on FASTER and compare it to Grace. As shown in Figure 2, the scaling performance of Horovod TensorFlow ResNet50 on the composable Faster cluster (cyan markers) is close to that of the Grace cluster (yellow diamond markers) for 2 GPUs on a single node. We next test several configurations of GPUs on nodes, gradually increasing the number of GPUs on each composed node. In general, the image throughputs of 1-node, 2-nodes, and 4-node

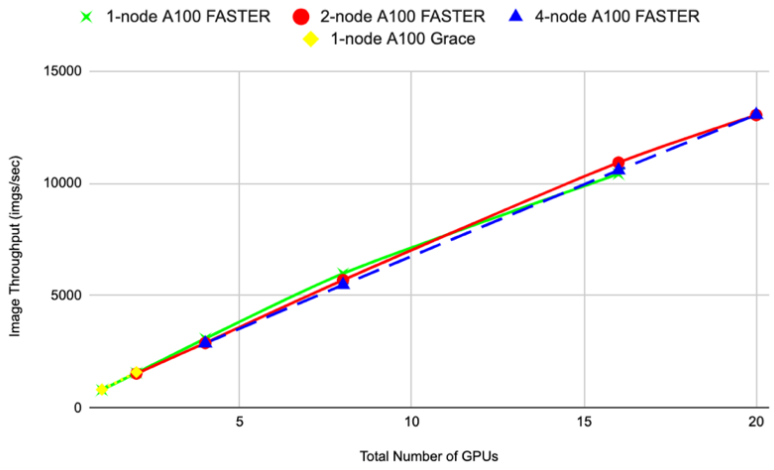

Figure 2: Horovod TensorFlow ResNet50 scaling on the FASTER and Grace supercomputing clusters

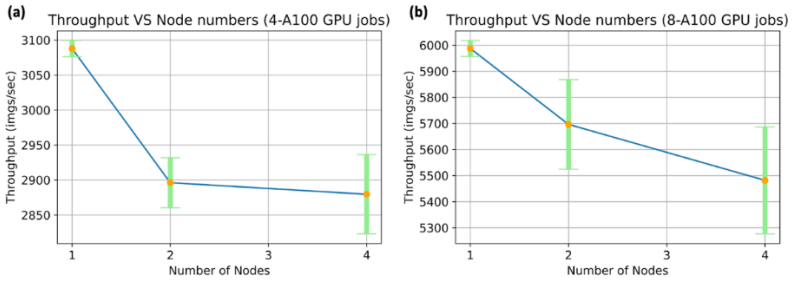

Figure 3: Impact of number of nodes composed with A100 GPUs on the image throughput.

jobs have similar scaling behaviors, all increasing almost linearly with the number of GPUs for up to 20 GPUs on a single node.

We next evaluate the communication overhead introduced by spreading the workload on FASTER across more composed nodes while maintaining the same total number of GPUs. Figure 3 shows that as the number of composed nodes increases the total image throughput of the used GPUs decreases. This is because more communication overhead occurs among the nodes as the number of nodes involved in a job increases. Figure 3(a) shows the results from experiments conducted with 1 node containing 4 A100s, 2 nodes each with 2 A100s, and 4 nodes each with 1 A100. An average value calculated from five iterations of each experiment is presented along with the associated error bars and standard deviation. Similarly, Figure 3(b) shows the values obtained from experiments performed with 1 node containing 8 A100s, 2 nodes each with 4 A100s, and 4 nodes each with 2 A100s. We observed a similar trend, with the image throughput reducing as the GPUs were distributed across a greater number of composed nodes.

To establish that these characteristics were not unique to the A100 GPUs, we performed a similar experiment using the on FASTER nodes composed with T4 GPUs. Figure 4(a) shows the results from experiments were conducted with 1 node containing 4 T4s, 2 nodes each with 2 T4s, and 4 nodes each with 1 T4 and Figure 4(b) shows that experiments were conducted with 1 node containing 8 T4s, 2 nodes each with 4 T4s, and 4 nodes each with 2 T4s. Unlike the case with A100 GPUs that found decreasing throughput with the increasing number of nodes, here we find that the performance remains consistent across different distributions of GPUs over changing number of composed nodes. This is likely because T4 GPUs have less computing power, and a longer computing overhead compared to A100 GPUs. It is possible that these factors play a larger role than the effect of the communication on the T4 composed nodes on FASTER.

Building on these findings, we next established the effect of distributing GPUs on a single composed node vs. multiple composed nodes for cases with 4 and 8 GPUs on a single composed node. We performed the same calculation, and the results are shown in Figure 5. To establish a baseline for this comparison, we used the throughput for a node with 4 A100 GPUs and 8 A100 GPUs and rated it at 100%. By now distributing these GPUs from 1 composed node to 2 composed nodes and 4 composed nodes, we find that the performance of jobs requiring 4 A100 GPUs decreases by 8.1% when the GPUs are composed on two nodes, and by 13.3% when the 4 A100 GPUs are composed on four nodes. Similarly, the performance of jobs requiring 8 A100 GPUs also decreases as we distribute them

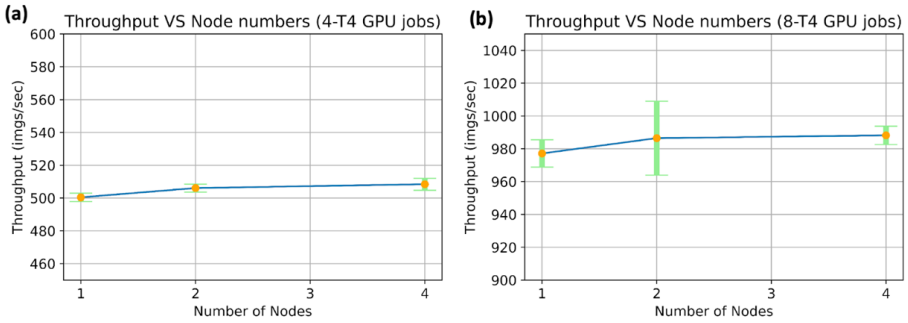

Figure 4: Impact of number of nodes composed with T4 GPUs on the image throughput.

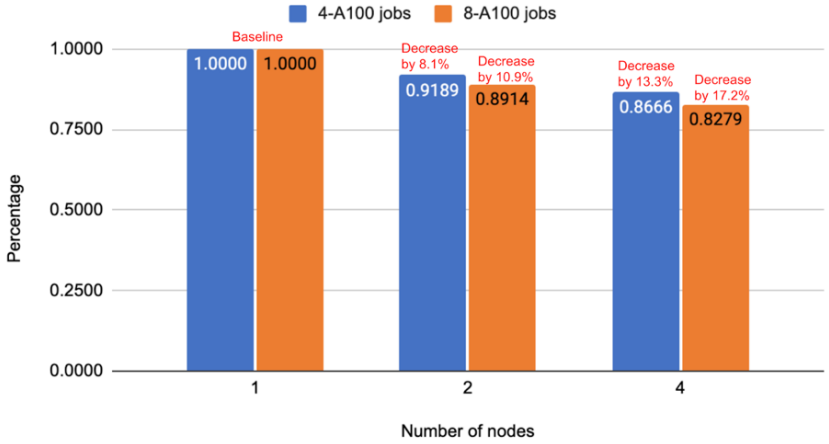

Figure 5: Performance drop due to inter-node communication.

over a larger number of composed nodes. In this case, we see an even larger performance drop.

In light of our findings from the presented Horovod TensorFlow ResNet50 model runs, one might expect that its best to compose nodes with the largest possible number of GPUs. Not only would such an approach seemingly guarantee better performance, but it would also reduce the amount of time system administrators spend configuring the machine. We are, however, cognizant that not all computing tasks need every GPU on a composed node. It is possible for a task to use only 1 GPU on a composed node that has 16 GPUs. To establish how these tasks would fare, we next compared the performance of using part or all the GPUs on a composed node on FASTER using the Horovod TensorFlow and PyTorch ResNet50 models. Four nodes on the FASTER cluster were composed with 4, 8, 12 and 16 A100 GPUs respectively. The results from performing these calculations using a sub-part of the total GPUs on each composed node allowed us to establish an estimate of performance expectations. Each subplot of Figure 6 represents job profiles where a selected number of A100 GPUs were used on these composed nodes. For example, Figure 6(a) shows the throughput of jobs performed on 1 A100 GPU on each of these composed nodes. As shown in Figures 6(a) and 6(b), the performance of FASTER is close to that of Grace. In Grace, GPUs are directly dedicated to the compute nodes while in FASTER the GPUs need to be composed to the nodes through shared PCIe connections which could cause a bottleneck.

The Coefficient of Variation (CV) was calculated for each case (subplot) and represents the proportion of the standard deviation to the mean and illustrates the level of variation in relation to the mean of the image throughput. The mathematical formula for coefficient variation is as follows.

$$CV = \frac{\sigma}{\mu}$$

Where σ is the standard deviation of the throughputs in each case and μ is the mean of the throughputs. The CV values, as observed in the subplots, are all below 0.7% for Horovod TensorFlow ResNet50 model experiments, which demonstrates that the performance variations are minimal when the same number of GPUs are utilized on nodes with different configurations. The CV values for Horovod PyTorch ResNet50 model experiments were higher compared to the Horovod TensorFlow ResNet50 model experiments, indicating greater performance variation. However, the CV values of all the cases were all below 2.7%, which is also small.

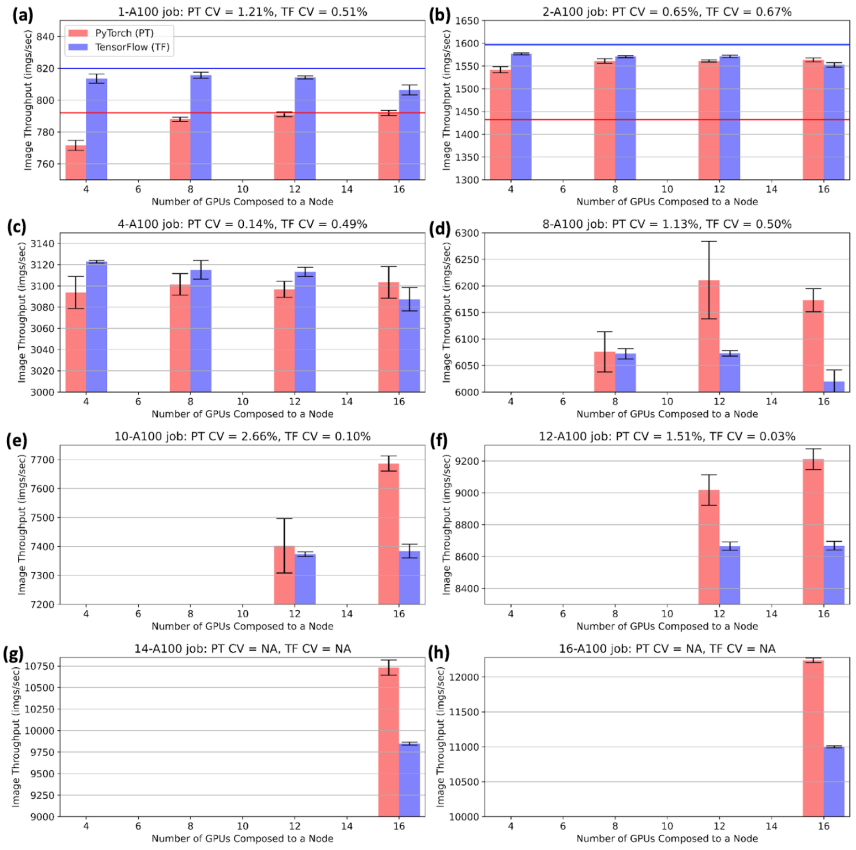

Figure 6: The performance of the Horovod TensorFlow (red bars) and PyTorch (purple bars) ResNet50 model on the FASTER nodes with differing numbers of GPUs composed. The blue and red lines in subplots (a) and (b) are performance data for TensorFlow and PyTorch on Grace, respectively.

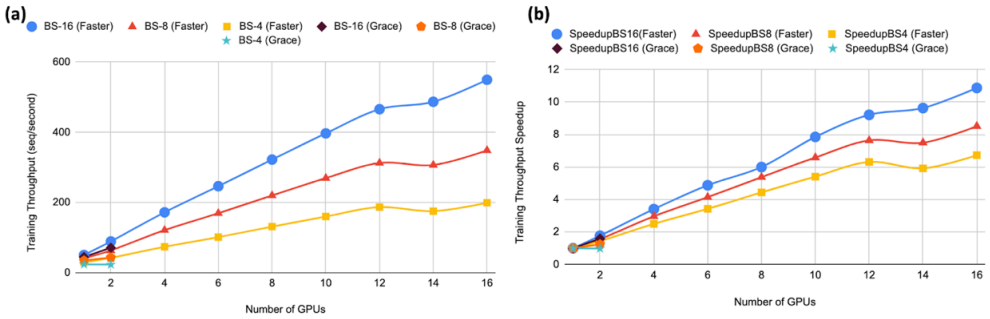

Figure 7: (a) The BERT-Large scaling behavior and (b) throughout speedup with different batch sizes (BS) as number of GPUs increase.

3.2 Fine-tuning of a Pre-trained PyTorch BERT-Large Language Model

Batch size (BS) and varying the precision of the model is known to impact performance as well. To understand how these factor impacts runs on CDIs, we fine-tuned a pre-trained PyTorch BERT-Large language model on the SQuAD dataset. Here, we calculated the training throughput on runs with increasing batch sizes {4, 8, and 16} for an increasing number of A100 GPUs composed to a single node with TF32 precision.

3.2.1 Batch Scaling. Our benchmark experiment results were obtained from an average of five iterations and are presented in Figure 7. For example, BS-16 indicates a batch size of 16 is used.

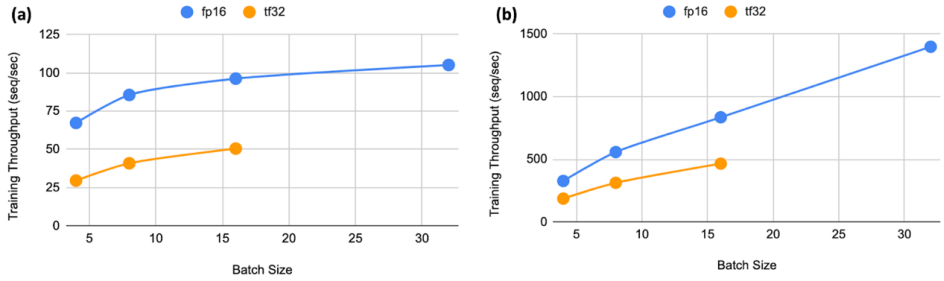

Figure 8: Training Throughput Comparison for FP16 and TF32 precisions on (a) a single A100 GPU and (b) 12 A100 GPUs.

Figure 7(a) demonstrates that the training sequence throughput increases almost linearly as the number of GPUs increases up to 16, with this performance increase being consistent across all batch sizes. BERT-Large, which has 340M parameters, requires significant compute and memory for pre-training and fine-tuning tasks. When utilizing multi-GPU distributed training for fine-tuning, BERT-Large utilizes PyTorch DataParallel and DistributedDataParallel training to distribute data across multiple GPUs and work on training sequences concurrently. Figure 7(b) depicts the speed-up for training throughput during BERT-Large fine-tuning. It shows that larger batch sizes lead to higher speed-ups for throughput, as more text sequences can be processed simultaneously, allowing for better hardware utilization. The A100 nodes used in our experiments have 40GB memory and can accommodate more data in their memory. Larger batch sizes result in more efficient GPU utilization in this case. Moreover, in this study, we found that there was a slight decrease in the performance when we ran the model on 14 GPUs as compared to running it on 12 GPUs using a batch size of 8 and a batch size of 4. On 16 GPUs, we again recorded an increase in the throughput performance. It is likely that the PCIe connections shared with the GPU could be a bottleneck in communications among other likely scenarios.

3.2.2 Precision Study. Noting that GPUs can perform calculations at different precision levels, we next performed a BERT-Large fine-tuning experiment using FP16 and TF32 precisions. We also compared the training throughput on a single A100 GPU and 12 A100 GPUs on a node composed with 12 A100 GPUs.

The results from these runs are shown in Figure 8. We find that the training throughput increases with batch size for both FP16 and TF32 precisions. In addition, FP16 precision exhibits higher training throughput than TF32 precision for the same batch size on a single GPU as well as for 12 GPUs. The speedup for FP16 over TF32 for batch size {4,8,16} is 2.27, 2.09 and 1.9 respectively on a single GPU and 1.75, 1.78 and 1.79 respectively for multi-GPU (12 GPUs). This is due to the fact that FP16 precision uses 16 bits of memory, which is half of the 32 bits used by TF32 precision, resulting in less memory bandwidth required to transfer FP16 data during training. Additionally, half precision arithmetic operations require fewer clock cycles to perform the same operation compared to full precision, resulting in faster operations and ultimately, faster training throughput. For batch size 32, we only have data points for FP16 precision as data does not fit in the memory for TF32 precision at this batch size.

4 CONCLUSIONS

The performance analysis for AI/ML workloads on the FASTER composable computing cluster and Grace traditional computing cluster show that the composable HPC layout, such as the FASTER cluster, can offer more flexibility by dynamically provisioning and reallocating resources. This could help meet the needs of diverse workloads that are common in HPC settings. We note that composing GPU-enabled nodes impacts scalability, though performance is similar to that observed in the common CPU-GPU HPC layout. The TensorFlow ResNet50 model in the Horovod distributed framework showed good scaling characteristics in both single-node and multi-node settings on the FASTER composable cluster. The PyTorch BERT-Large model too exhibited good scaling behavior in single-node multi-GPU settings on the FASTER composable cluster. The image throughput increased with the increasing number of GPUs and had similar scaling behaviors in the single-node and multi-node settings. The effect on performance by distributing the same number of GPUs across a greater number of composable nodes, and different node configurations was quantified. The performance variations are minimal when the same number of GPUs are utilized on nodes with different configurations, i.e., composed of different numbers of GPUs. In an environment running similar workloads, system administrators could choose specific composed configurations of CDIs without researchers losing out on performance. We, however, note that the performance characteristics of different types of GPUs - A100 and T4 differed when running similar workloads. The balance between communication overhead, computing speed and memory space should be considered while choosing the appropriate GPUs.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) award number 2112356 ACES - Accelerating Computing for Emerging Sciences, NSF award number 1925764 SWEETER - SouthWest Expertise in Expanding, Training, Education and Research, NSF award number 2019136 BRICCs - Building Innovation at Community Colleges, NSF award number 2019129 FASTER - Fostering Accelerated Scientific Transformations, Education, and Research,

staff and students at Texas A&M High Performance Research Computing including Shaina Le, Wesley Brashear, Abhinand Nasari, and Ritika Mendjoge.

REFERENCES

[1] National Artificial Intelligence Research Resource Task Force Report: Strengthening and Democratizing the U.S. Artificial Intelligence Innovation Ecosystem - An Implementation Plan for a National Artificial Intelligence Research Resource. Retrieved March 3, 2023 from https://www.ai.gov/wp-content/uploads/2023/01/NAIRR-TF-Final-Report-2023.pdf~

[2] Report to the President on Computational Science: Ensuring America's Competitiveness, The President's Information Technology Advisory Committee (PITAC), June 2005.

[3] T. Hey, S. Tansley, K. Tolle, The Fourth Paradigm: Data-Intensive Scientific Discovery, October 16, 2009.

[4] Most-detailed-ever simulations of black hole solve longstanding mystery. Retrieved March 3, 2023 from https://phys.org/news/2019-06-most-detailed-ever-simulations-black-hole-longstanding.html

[5] Robert A. Dick, Kaneil K. Zadrozny, Chaoyi Xu, Florian K. M. Schur, Terri D. Lyddon, Clifton L. Ricana, Jonathan M. Wagner, Juan R. Perilla, Barbie K. Ganser-Pornillos, Marc C. Johnson, Owen Pornillos & Volker M. Vogt Inositol phosphates are assembly co-factors for HIV-1 Nature volume 560, pages 509–512(2018). https://doi.org/10.1038/s41586-018-0396-4

[6] Laser Interferometer Gravitational-Wave Observatory (LIGO) - Computation and Data Collection. Retrieved on March 3, 2023 from https://www.ligo.caltech.edu/page/ligo-technology

[7] "1st Workshop on Composable Systems (COMPSYS 2022)," in 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lyon, France, 2022 pp. 1192-1193. doi: https://doi.org/10.1109/IPDPSW55747.2022.00204~

[8] I-Hsin Chung, Bulent Abali, and Paul Crumley. 2018. Towards a Composable Computer System. In Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2018). Association for Computing Machinery, New York, NY, USA, 137–147. https://doi.org/10.1145/3149457.3149466

[9] HPCwire. 2022. ACES 'Composable' Supercomputer Gets Ready for Phase One Use. Retrieved March 3, 2023 from https://www.hpcwire.com/2022/04/04/aces-composable-supercomputer-gets-ready-for-phase-one-use/

[10] HPCwire. 2022. SDSC and GigaIO to Highlight $11.25M Composable Cyberinfrastructure Ecosystem at SC22. Retrieved March 3, 2023 from https://www.hpcwire.com/off-the-wire/sdsc-and-gigaio-to-highlight-11-25m-composable-cyberinfrastructure-ecosystem-at-sc22/

[11] Abhinand S. Nasari, Richard Lawrence, Zhenhua He, Hieu Le, Mario Michael Krell, Alex Tsyplikhin, Mahidhar Tateneni, Tim Cockerill, Lisa M. Perez, Dhruva K. Chakravorty, and Honggao Liu. (2022). Benchmarking the Performance of Accelerators on National Cyberinfrastructure Resources for Artificial Intelligence/Machine Learning Workloads. In Practice and Experience in Advanced Research Computing, pp. 1-9. 2022. https://dl.acm.org/doi/10.1145/3491418.3530772~

[12] National Research Platform (NRP). Retrieved April 11, 2023 from https://nationalresearchplatform.org/

[13] Nasari, Abhinand, Hieu Le, Richard Lawrence, Zhenhua He, Xin Yang, Mario Krell, Alex Tsyplikhin *et al.* 2022. Benchmarking the Performance of Accelerators on National Cyberinfrastructure Resources for Artificial Intelligence/Machine Learning Workloads. In Practice and Experience in Advanced Research Computing, pp. 1-9. 2022. https://doi.org/10.1145/3491418.3530772

[14] HPCwire. 2022. Using Composable Infrastructure to Get More ROI from Clusters. Retrieved March 3, 2023 from https://www.hpcwire.com/2022/04/18/using-composable-infrastructure-to-get-more-roi-from-clusters-2/

[15] Texas A&M High Performance Research Computing. Retrieved March 3, 2023 from https://hprc.tamu.edu/faster/

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 770–778. https://doi.org/10.1109/CVPR.2016.90~

[17] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv: 1810.04805 Retrieved March 3, 2023 from https://doi.org/10.48550/arXiv.1810.04805~

[18] NVIDIA. 2020. A100 40GB PCIe Product Brief. (September 2020). Retrieved March 3, 2023 from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/A100-PCIE-Prduct-Brief.pdf

[19] NVIDIA. 2020. T4 70W LOW PROFILE PCIe GPU ACCELERATOR. Product Brief. (September 2020) Retrieved March 3, 2023 from https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-product-brief.pdf

[20] National Science Foundation Expanse. Retrieved April 11, 2023 from https://www.sdsc.edu/services/hpc/expanse/

[21] National Science Foundation Anvil. Retrieved April 11, 2023 from https://www.nsf.gov/awardsearch/showAward?AWD_ID$=$2005632

[22] National Science Foundation Delta. Retrieved April 11, 2023 from https://www.ncsa.illinois.edu/research/project-highlights/delta/

[23] Texas A&M High Performance Research Computing, Introduction to Grace. Retrieved March 3, 2023 from https://hprc.tamu.edu/wiki/Grace:Intro

[24] Horovod. Retrieved June 16, 2023 from https://github.com/horovod/horovod

[25] Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019).

[26] Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv preprint arXiv:1910.01108 (2019).

[27] Jiao, Xiaoqi, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. "Tinybert: Distilling bert for natural language understanding." arXiv preprint arXiv:1909.10351 (2019).

[28] TensorFlow Benchmarks GitHub repository. Retrieved March 3, 2023 from https://github.com/tensorflow/benchmarks

[29] Horovod PyTorch Benchmark GitHub repository. Retrieved March 3, 2023 from https://github.com/horovod/horovod

[30] NVIDIA Deep Learning Examples. Retrieved March 3, 2023 from https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT

[31] NVIDIA. 2022. NGC catalog. Retrieved March 3, 2023 from https://catalog.ngc.nvidia.com/containers

[32] NVIDIA HPC-Benchmarks, Retrieved April 21, 2023 from https://catalog.ngc.nvidia.com/orgs/nvidia/containers/hpc-benchmarks