Comparative Analysis of Standard Image Classification Model Training Techniques with FSDP

Aditya Mohan
Arizona State University
Tempe, Arizona, USA
amohan62@asu.edu

Baibhav Phukan
Arizona State University
Tempe, Arizona, USA
bphukan@asu.edu

Nagaraju Machavarapu
Arizona State University
Tempe, Arizona, USA
nmachav1@asu.edu

ABSTRACT

As the demand for larger AI models grows, training such models becomes increasingly challenging due to computational constraints. PyTorch's Fully Sharded Data Parallel (FSDP) presents a promising solution by distributing model parameters across multiple GPUs, thereby enabling the training of massive models that wouldn't fit on a single GPU. This study provides a comprehensive comparative analysis of standard image classification model training techniques with FSDP. The primary objective is to assess the effectiveness of FSDP in enhancing the training process for large-scale models. The study compares FSDP against traditional techniques using MobileNetV2 and ResNet on the CIFAR-10 dataset, exploring its scalability, efficiency, and practical implications. Through these experiments and analysis, this study provides insights into the strengths and weaknesses of FSDP, contributing to the advancement of distributed training frameworks.

CCS CONCEPTS

• **Machine Learning** → *Image Classification*; • **PyTorch** → *Fully Sharded Data Parallel*.

KEYWORDS

FSDP, GPU, MobileNet, ResNet, CIFAR-10, Machine Learning, Neural Networks

1 INTRODUCTION

According to recent studies, larger models have been shown to enhance AI performance, with parameters exploding 10,000-fold in just 3 years. However, training such huge models is difficult due to the massive computing power and complex software needed. PyTorch, a popular AI framework, is building tools to address this challenge. Notably, "Distributed data parallelism" is a common technique, but it limits models to single GPUs. Newer methods like DeepSpeed ZeRO and FairScale's Sharded Data Parallel break this barrier by splitting the model across multiple GPUs, simplifying the training process for massive AI models.

PyTorch's Fully Sharded Data Parallel (FSDP) is an advanced technique for training extremely large AI models that would not fit on a single GPU. FSDP cuts the model into smaller pieces called "shards" and spreads them across multiple GPUs. Here is how it works:

Sharding: FSDP flattens and splits the model's parameters, gradients, and optimizer states into manageable chunks.

Distributed training: Each GPU only works on its assigned shards, reducing memory usage per device.

Communication: GPUs talk to each other to exchange information needed for calculations and updates.

Assembly: FSDP automatically combines the results from all GPUs to obtain the final model updates.

To address the challenges posed by training increasingly larger AI models, this study proposes to conduct a comprehensive comparative analysis of standard image classification model training techniques with PyTorch's Fully Sharded Data Parallel (FSDP). The primary objective is to assess the effectiveness of FSDP in enhancing the training process for large-scale models that wouldn't fit on a single GPU. This analysis is intriguing for several reasons:

Scalability: As AI models continue to grow in size, scalability becomes a critical concern. Techniques like FSDP offer a solution by distributing the computational load across multiple GPUs, potentially enabling the training of even larger models.

Efficiency: By reducing memory usage per device and optimizing communication between GPUs, FSDP aims to make the training of massive models more efficient, thereby lowering the computational resources required.

Practical Implications: Understanding the practical implications of adopting FSDP for large-scale model training can influence decision-making processes regarding framework selection, infrastructure requirements, and training methodologies.

Comparative Analysis: Conducting a comparative analysis allows for a systematic evaluation of FSDP against existing techniques. By benchmarking FSDP against traditional methods, there is scope to identify its strengths, weaknesses, and potential areas for improvement, thereby contributing to the advancement of distributed training frameworks.

2 RELATED WORK

In the challenge of training ever-larger AI models, PyTorch's Fully Sharded Data Parallel (FSDP) stands tall amidst a landscape of data parallelism techniques. While traditional Data Parallelism (DP) excels in its simplicity and robustness [6], its single-GPU limitation hinders its application to larger models. Gradient Sharded Data Parallelism (GSDP) [4] attempts to overcome this hurdle by splitting gradients across GPUs, but parameter sharding remains a challenge. Here, FSDP shines by fragmenting both parameters and gradients, allowing near-linear scalability observed in research [3].

Compared to model parallelism approaches like DeepSpeed ZeRO [5], FSDP offers a hybrid advantage. While ZeRO achieves impressive memory reduction, it can introduce complex communication patterns and overhead [2]. FSDP strikes a balance by retaining data parallelism's efficient communication while enabling the training of models larger than ZeRO can handle.

However, FSDP is a fairly new contender still under active development. Its integration with other techniques like pipeline parallelism requires further exploration [1]. Additionally, its nascent

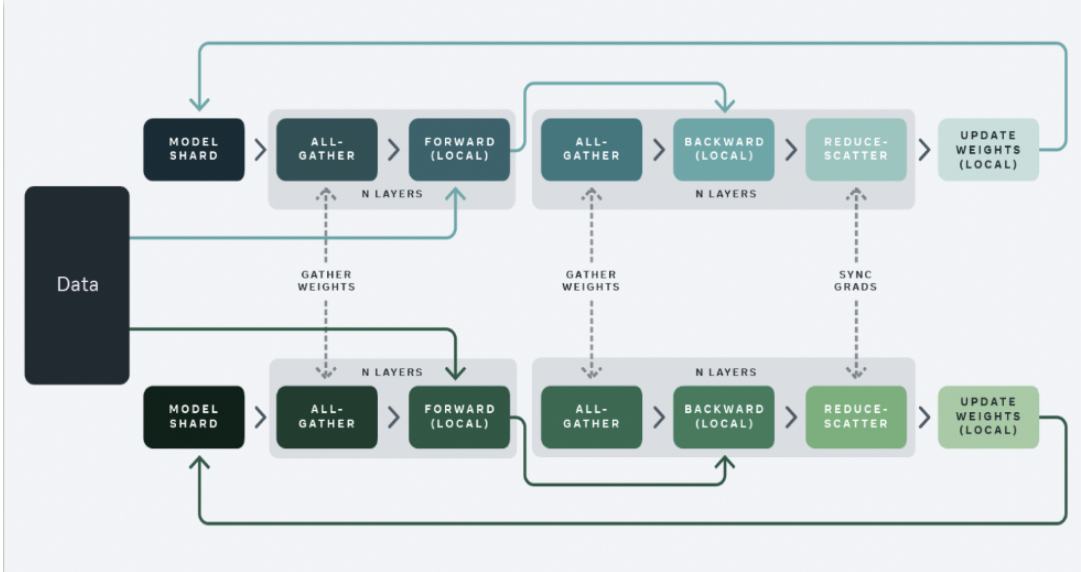

Figure 1: The FSDP pipeline

stage necessitates careful configuration and coding expertise for optimal performance [3].

Despite these considerations, FSDP's potential is undeniable. Studies showcase its success in training LLMs like Megatron-Turing NLG [7], demonstrating its effectiveness in pushing the boundaries of NLP tasks. Investigations into its applicability in other domains like computer vision are underway, hinting at its broader impact.

3 EXPERIMENTAL SETUP

3.1 Method

This study implements and compares FSDP against traditional training techniques on a standard image classification task. It's focused on two widely known pre-trained models: MobileNetV2 and ResNet, both of which are available in the torchvision library. These models were chosen to explore the impact of FSDP across a spectrum of model complexities.

These models were chosen specifically because of their contrasting nature - MobileNetV2 is a lightweight architecture and ResNet is known for its ability to handle deep architectures. A broad spectrum of architectures would lead to a more robust evaluation.

MobileNetV2 is a lightweight convolutional neural network designed for efficiency on mobile devices and resource-constrained environments. It balances accuracy with speed by using depthwise separable convolutions and linear bottleneck layers.

ResNet, on the other hand, is a powerful convolutional neural network known for its accuracy and ability to handle deep architectures. It utilizes skip connections to address the vanishing gradient problem, allowing for better information flow and training deeper networks.

3.2 Data

To conduct these experiments, the CIFAR-10 dataset is used as it is a widely used benchmark dataset for image classification tasks. It consists of 60,000 32*32 color images, with 6000 images per class. The dataset is split into 50,000 training images and 10,000 test images. The dataset covers a diverse range of objects and animals commonly found in everyday environments, making it suitable for our analysis. CIFAR-10 is a sweet spot for both MobileNetV2 and ResNet: its 32x32 images are perfect for their efficient designs, its 10 classes offer a good balance of learning challenge and manageability, and its widespread use allows for easy performance comparison.

3.3 Environment Setup

In this study, the proposed experiments were performed by leveraging ASU Research Computing Supercomputer Sol's multi-GPU availability and computational power. PyTorch's Fully Sharded Data Parallel was integrated into the training pipeline for both MobileNetV2 and ResNet models. This involved sharding the model parameters, gradients, and optimizer states, enabling distributed training across multiple GPUs.

The experiments included training both models with and without using FSDP. For the baseline, traditional model training techniques using a single GPU instance were employed. The key parameters such as training speed, memory usage, time taken, per-epoch time and accuracy were closely monitored and compared between the two approaches.

4 RESULTS

This comprehensive study provides various insights into the performance and impact of FSDP on training large models, specifically MobileNetV2 and ResNet on CIFAR-10 dataset. The Fully Sharded Data Parallelism (FSDP) technique showcases remarkable efficiency in memory utilization, especially notable when employed with the ResNet18 architecture. These experiments reveal a substantial reduction in peak GPU memory consumption, plummeting to a mere 0.27 GB with FSDP implementation. While FSDP does

Table 1: Comparison of Results Across Configurations

Metric	*Baseline with 1 GPU*	*Baseling with 2 GPUs*	*FSDP with 2 GPUs*
GPU Memory Utilization			
ResNet-18	0.13 GB	0.13 GB	GPU0: **0.04 GB**
MobileNet-v2	0.03 GB	0.03 GB	GPU0: **0.03 GB**
GPU Peak Memory			
ResNet-18	0.74 GB	0.74 GB	GPU0: **0.27 GB**
MobileNet-v2	0.24 GB	0.24 GB	GPU0: **0.21 GB**
Average Time Per Epoch (sec)			
ResNet-18	**30.23**	44.07	32
MobileNet-v2	**34.12**	48.46	35
Total Time Taken (sec)			
ResNet-18	**906.825**	1337.46	1005.49
MobileNet-v2	**1023.64**	1453.754	1135.67
Accuracy (%)			
ResNet-18	71.06	69.16	**78.8**
MobileNet-v2	52	50	**53.6**

not inherently bolster model accuracy, it consistently maintains performance at a competitive level. Moreover, integrating FSDP into our framework yields a noteworthy reduction in total execution time, contributing to accelerated model training and inference processes compared to configurations without FSDP. Across these evaluations, FSDP emerges as the top performer, demonstrating its efficacy in optimizing both training and inference workflows. Notably, the results underscore that the Baseline model with one GPU achieves the next best performance, followed by the Baseline with two GPUs. These findings emphasize that merely adding more GPUs does not guarantee a linear improvement in performance, as factors such as communication overhead and parallel processing inefficiencies come into play. Consequently, employing sharding techniques proves to be a judicious choice in enhancing overall system efficiency.

5 CONCLUSION

In conclusion, this study has provided a comprehensive analysis of PyTorch's Fully Sharded Data Parallel (FSDP) technique in comparison to traditional image classification model training techniques. Through experiments conducted on the CIFAR-10 dataset using MobileNetV2 and ResNet architectures, several key findings have emerged. Firstly, FSDP demonstrates remarkable efficiency in memory utilization and peak GPU memory consumption. While FSDP does not inherently improve model accuracy, it consistently maintains competitive performance levels. Moreover, integrating FSDP into the training framework leads to notable reductions in total execution time, accelerating both training and inference processes. These results underscore the efficacy of FSDP in optimizing training workflows for large-scale models that wouldn't fit on a single GPU.

6 FUTURE WORK

Despite the promising findings of this study, there remain several avenues for future research and development. Firstly, further exploration is warranted into the integration of FSDP with other parallelism techniques, such as pipeline parallelism, to enhance its scalability and efficiency further. Additionally, investigating FSDP's applicability across different domains beyond image classification, such as natural language processing (NLP) and reinforcement learning, could unveil its broader impact and potential. Furthermore, conducting longitudinal studies to assess FSDP's performance and scalability as model sizes continue to grow would provide valuable insights into its long-term effectiveness. Overall, the continued refinement and extension of FSDP hold promise for advancing distributed training frameworks and facilitating the training of increasingly complex AI models.

REFERENCES

[1] Aaron Archer, Matthew Fahrbach, Kuikui Liu, and Prakash Prabhu. 2023. Pipeline Parallelism for DNN Inference with Practical Performance Guarantees. arXiv:2311.03703 [cs.LG]

[2] A. Babsha, Y. You, M. Li, J. Joneja, and Y. Huang. 2023. DeepSpeed: System Design for Large Model Training. (2023). arXiv:2301.02691 [cs.DC]

[3] Y. Huang, S. Li, Y. Cheng, C. Bao, S. Li, L. Chen, and et al. 2023. Fully Sharded Data Parallel (FSDP) for Efficient Large Model Training on GPUs. (2023). arXiv:2304.11277 [cs.LG]

[4] Kabir Nagrecha. 2023. Systems for Parallel and Distributed Large-Model Deep Learning Training. arXiv:2301.02691 [cs.DC]

[5] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. arXiv:1910.02054 [cs.LG]

[6] Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. 2019. Measuring the Effects of Data Parallelism on Neural Network Training. arXiv:1811.03600 [cs.LG]

[7] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. arXiv:2201.11990 [cs.CL]