Indian Institute of Technology Kanpur

SURGE INTERNSHIP REPORT

"Clustering GPU for Multinode Training "

Student Name	Student ID
Abhishek Maurya	2430398
Abhishek Sahu	2430384

Guide:

Dr. Tushar Sandhan

1 Introduction

In recent years, artificial intelligence and machine learning have become really important, and having strong computers to run these models is crucial. GPUs (Graphics Processing Units) are especially good at this because they can handle many tasks at once. This project looks at why GPUs are necessary for running machine learning models and explores the way to use them in PyTorch: Distributed Data Parallel (DDP).

By systematically exploring the applications of DDP, this study provides insights into their efficiency and effectiveness in various computational settings. Through a series of experiments, we compare the performance of models trained on CPUs, single GPUs, and multiple GPUs in both single-node and multi-node environments. The findings of this project highlight the significant speedup provided by GPUs over CPUs and the complexities associated with scaling training across multiple nodes, emphasizing the need for optimized gradient communication strategies. By training a ResNet-152 model for image classification, this project not only illustrates the potential of DDP in enhancing deep learning workflows but also identifies key challenges and areas for further research to maximize their practical utility.

2 Objectives

The main objective of this project is to create a GPU cluster, a group of computers each equipped with a GPU, that work together to train machine learning models. The specific goals include:

1. **Cluster Formation**: Connecting three computers, each with its own GPU, to form a cohesive GPU cluster. This cluster aims to leverage the combined computational power of multiple GPUs, significantly speeding up the training process and improving overall efficiency.
2. **Centralized Data Management and Seamless Communication**: Implementing a Network File System (NFS) shared directories to centralize data storage, ensuring all nodes in the cluster have access to a common data repository and eliminating data redundancy. Additionally, establish passwordless SSH between the cluster nodes to facilitate seamless and secure communication, enabling efficient coordination and data transfer between nodes.

3 Cluster Architecture

3.1 Compute Nodes

These are the computers within the cluster that do the work. Each computer, called a node, has a processor (CPU), memory (RAM), and often a Graphics Processing Unit (GPU) for faster training in deep learning.
Cluster we used:

- Two nodes with a single NVIDIA GeForce RTX 3060 GPU each.

- Single node with two NVIDIA GeForce RTX 4080 GPUs each.

3.2 GPUs for Deep Learning

GPUs (Graphics Processing Units) are essential in deep learning because they significantly speed up the training process by handling many tasks simultaneously. Unlike CPUs, GPUs are optimized for parallel processing, making them ideal for computationally intensive operations like matrix multiplications. In our project, we used a GPU cluster with multi GPU-equipped computers to enhance training speed and efficiency. This setup demonstrated the importance of GPUs in modern deep learning, allowing us to handle larger datasets and more complex models effectively.

3.3 Networking Infrastructure

In this project, We used a multi-node setup connected via a Local Area Network (LAN). Each node was accessed and managed through SSH (Secure Shell) from personal computer. This allowed us to control and coordinate the processes on all nodes using the local network, ensuring efficient communication and synchronization between the nodes(refer Fig. 1).

Before beginning, ensure that your nodes can communicate over TCP/IP with the following steps:

1. Test the Ping command to verify network connectivity between nodes.
2. Add the hostname of the parent node to the '/etc/hosts' file on every child node to ensure hostname resolution.

   Example:

   ```
   sudo nano /etc/hosts
   ```

   Add the following line:

   ```
   <parent_node_IP_address> <parent_node_hostname>
   ```

   Replace `<parent_node_IP_address>` and `<parent_node_hostname>` with the actual IP address and hostname of the parent node.
3. Ensure proper socket configuration to facilitate communication between nodes.

These steps are crucial to establish and verify network connectivity and hostname resolution across your distributed environment.

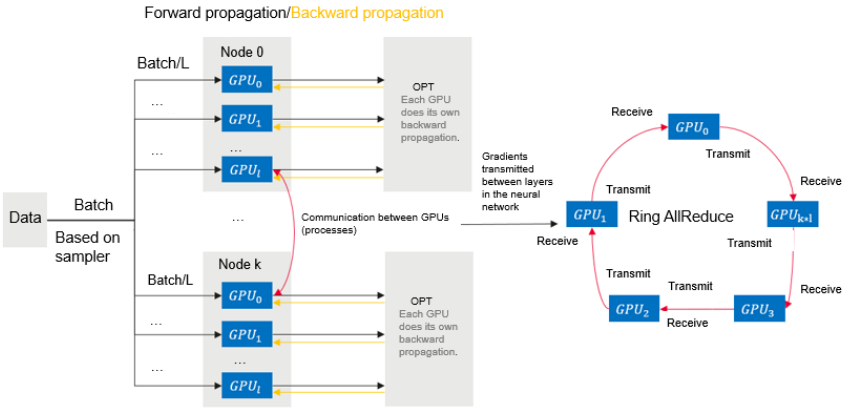

Figure 1: Cluster Backend

4 Specialized Configurations

4.1 GPUs with CUDA Support

In our project, GPUs with CUDA support were crucial for multi-node training. CUDA (Compute Unified Device Architecture) is a parallel computing platform and API model created by NVIDIA that allows GPUs to perform complex computations efficiently.

Even without using SLURM for job scheduling, we managed multi-node training by utilizing PyTorch libraries and local connections. The CUDA support allowed seamless execution of tensor operations across multiple GPUs, ensuring efficient gradient calculations and data transfers between nodes. This setup optimized our training process, highlighting the importance of CUDA-enabled GPUs in handling the computational demands of distributed deep learning.

5 Result And Observations

5.1 Batch-size 64

Setup	Total no. of GPUs	Total Training time
Single node	1	274.44 secs
Multi-GPU	2	236.74 secs
Multi-Node	4	1500.0 secs

5.2 Batch-size 128

Setup	Total no. of GPUs	Total Training time
Single node	1	210.58 secs
Multi-GPU	2	131.51 secs
Multi-Node	4	1059.38 secs

5.3 Batch-size 256

Setup	Total no. of GPUs	Total Training time
Single node	1	180.45 secs
Multi-GPU	2	80.25 secs
Multi-Node	4	547.88 secs

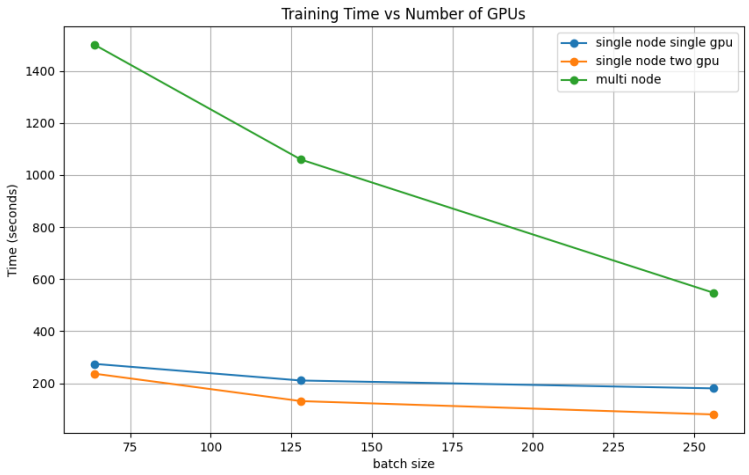

Figure 2: Performance Analysis

6 Discussion

In our experiments, we trained the model using different configurations: single-node with multiple GPUs, single-node with a single GPU, and multi-node training. We observed the following performance trends:

- Training with two GPUs on a single node outperformed training with a single GPU using torchrun.
- Training with a single GPU on a single node showed better performance compared to multi-node training.

The slower performance of multi-node training can be attributed to several factors:

- **Communication Overhead**: As nodes work in parallel, communicating and synchronizing their progress can introduce overhead. This needs to be optimized for efficient 5 training
- **Job Scheduling and Management** : Effectively scheduling and managing jobs across multiple nodes is essential for maximizing cluster utilization and avoiding bottlenecks.

7 Future Plan

To further improve the efficiency and performance of our multinode training setup, we plan to implement the following enhancements:

7.1 Building a SLURM Cluster

1. Building a SLURM cluster using cluster management software like Kubernetes.

2. Implementing scheduling and resource management software SLURM to enhance the cluster's scalability, flexibility, and efficiency by providing advanced job scheduling and resource allocation capabilities to better manage computational workloads.

7.2 High-Bandwidth Ethernet Setup

- **Increased Bandwidth:** Ensuring faster data transfer rates between nodes.
- **Lower Latency:** Reducing the time taken for data packets to travel between nodes, improving overall training efficiency.
- **Cost-Effectiveness:** Offering a more affordable alternative to InfiniBand while still meeting performance requirements.

These enhancements will significantly optimize our distributed training workflows, enabling us to handle larger datasets and more complex models with greater efficiency.

Cost Estimation for 5GbE Ethernet Setup

- **Ethernet Switch:** 5-port 5GbE switch - 17000 Rs
- **Network Interface Cards (NICs):** 5GbE NICs for 3 nodes - 8,000 Rs each$\times$ 3 = 24,000 Rs

8 References

[1] Github link: https://github.com/Abhisahu22/Surge_files.

[2] Guidance for building a gpu cluster using NVIDIA DeepOps : https://arxiv.org/pdf/2405.00030.

[3] Torchrun(Elastic launch) for faster multinode training : https://pytorch.org/docs/stable/elastic/run.html.

[4] Guidance for building a gpu cluster: https://youtu.be/KaAJtI1T2x4?si=MFdD2J85DYOd7McZ.