DE Project Report

Anvith Pendekatla

20MCME17

Contents

1 Introduction

This report documents the work done by me as a part of the project - **Speech Recognition using Federated Learning**.

As a part of understanding the project's problem statement, I went through multiple research papers understanding whisper (OpenAI LLM), PEFT and FedPEFT. I also spent a significant amount of time understanding the FLOWER framework that helps us in setting the Federated Learning environment.

2 Work Done

This section contains the specific work done by me that helped me learn more about the project's requirements, allowing me to contribute to the project.

2.1 Implementing an FL Setup using terminals as clients

This subsection demonstrates a Federated Learning environment using the FLOWER framework.

I have taken a model from tensorflow **keras** (open source neural networks library) and implemented the same in a federated learning setup.

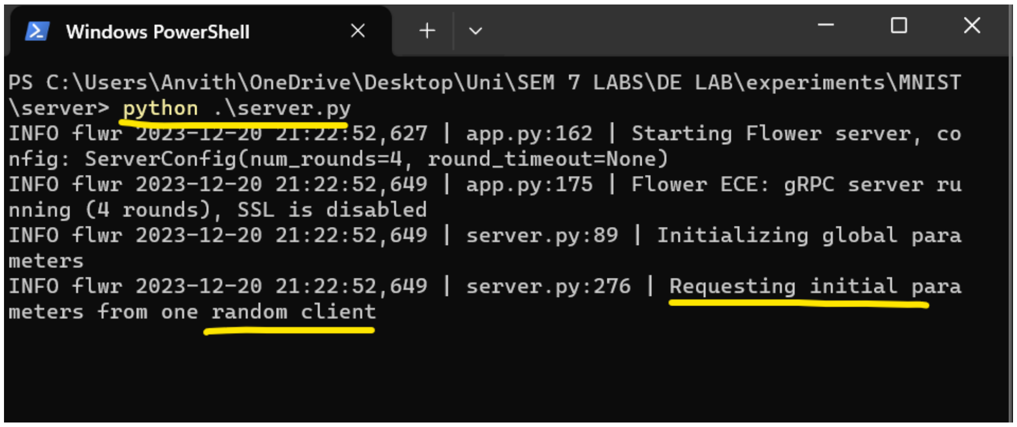
The **MNIST** dataset has been used to train this model. The task selected is *Image Classification.*

There are two clients involved in this Federated Learning process. These clients train local models on their non-IID subsets of the MNIST dataset. (Non-IID data distribution implies that the clients have different subsets of the data, ensuring a more realistic and challenging Federated Learning scenario)

The server coordinates the training process by aggregating model updates from clients and broadcasting the global model updates.

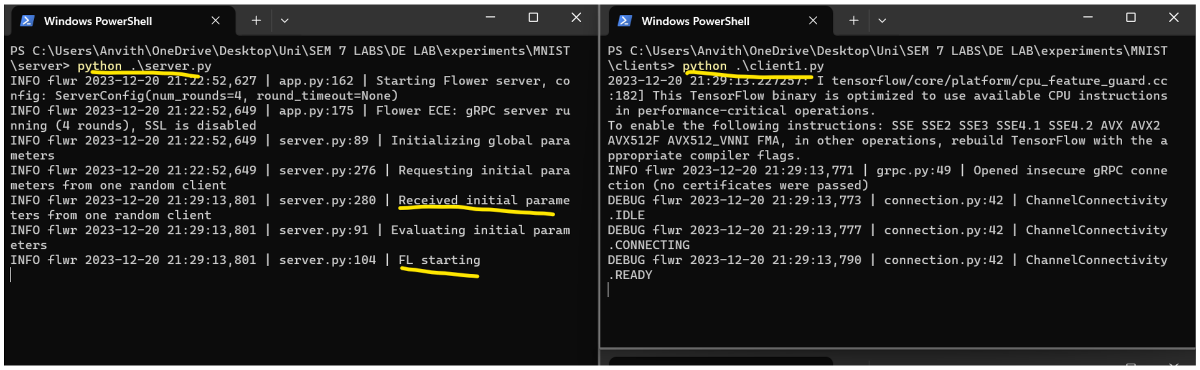
Below are the results of this experiment:

When the server starts running, it tries to initialize the global parameters. But since no client is running **yet**, it requests for initial parameters and waits for a random client.



```
PS C:\Users\Anvith\OneDrive\Desktop\Uni\SEM 7 LABS\DE LAB\experiments\MNIST
\server> python .\server.py
INFO flwr 2023-12-20 21:22:52,627 | app.py:162 | Starting Flower server, co
nfig: ServerConfig(num_rounds=4, round_timeout=None)
INFO flwr 2023-12-20 21:22:52,649 | app.py:175 | Flower ECE: gRPC server ru
nning (4 rounds), SSL is disabled
INFO flwr 2023-12-20 21:22:52,649 | server.py:89 | Initializing global para
meters
INFO flwr 2023-12-20 21:22:52,649 | server.py:276 | Requesting initial para
meters from one random client
```

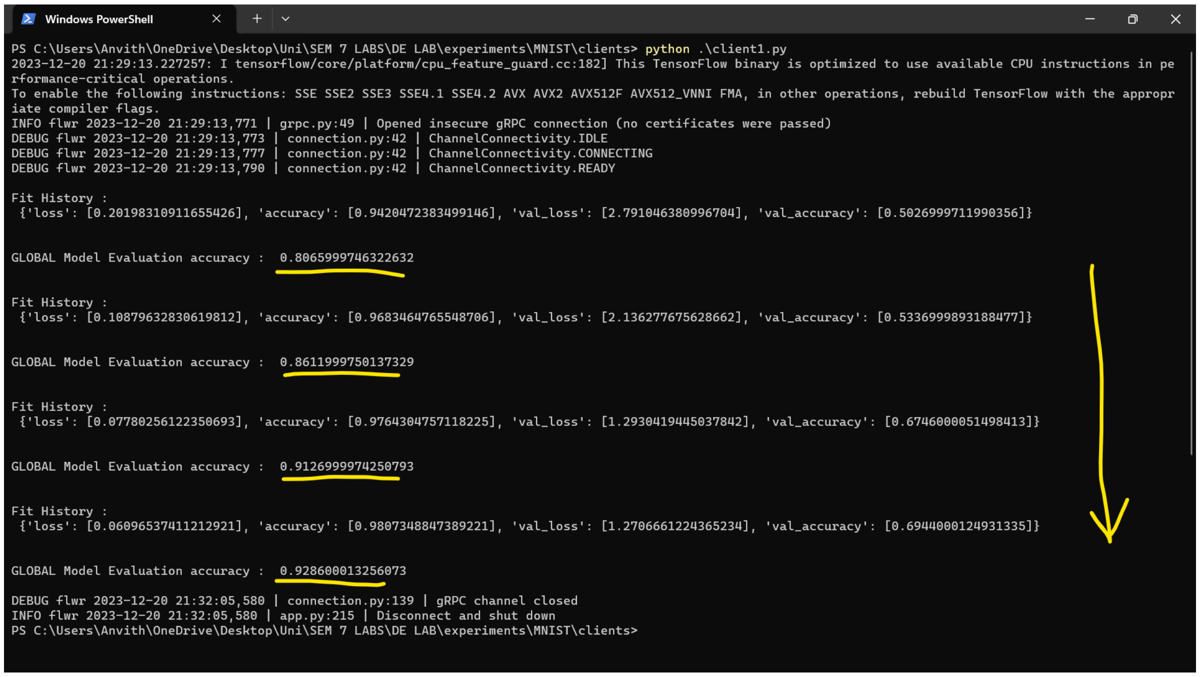
Now, after starting one client, the server receives the parameters and initiates the FL process.



```
PS C:\Users\Anvith\OneDrive\Desktop\Uni\SEM 7 LABS\DE LAB\experiments\MNIST
\server> python .\server.py
INFO flwr 2023-12-20 21:22:52,627 | app.py:162 | Starting Flower server, co
nfig: ServerConfig(num_rounds=4, round_timeout=None)
INFO flwr 2023-12-20 21:22:52,649 | app.py:175 | Flower ECE: gRPC server ru
nning (4 rounds), SSL is disabled
INFO flwr 2023-12-20 21:22:52,649 | server.py:89 | Initializing global para
meters
INFO flwr 2023-12-20 21:22:52,649 | server.py:276 | Requesting initial para
meters from one random client
INFO flwr 2023-12-20 21:29:13,801 | server.py:280 | Received initial parame
ters from one random client
INFO flwr 2023-12-20 21:29:13,801 | server.py:91 | Evaluating initial param
eters
INFO flwr 2023-12-20 21:29:13,801 | server.py:104 | FL starting
```

```
PS C:\Users\Anvith\OneDrive\Desktop\Uni\SEM 7 LABS\DE LAB\experiments\MNIST
\clients> python .\client1.py
2023-12-20 21:29:13.227257: I tensorflow/core/platform/cpu_feature_guard.cc
:182] This TensorFlow binary is optimized to use available CPU instructions
 in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2
AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the a
ppropriate compiler flags.
INFO flwr 2023-12-20 21:29:13,771 | grpc.py:49 | Opened insecure gRPC conne
ction (no certificates were passed)
DEBUG flwr 2023-12-20 21:29:13,773 | connection.py:42 | ChannelConnectivity
.IDLE
DEBUG flwr 2023-12-20 21:29:13,777 | connection.py:42 | ChannelConnectivity
.CONNECTING
DEBUG flwr 2023-12-20 21:29:13,790 | connection.py:42 | ChannelConnectivity
.READY
```

When more than 1 client is available, the FL process kicks off.
In *ServerConfig*, I have set the number of FL rounds to 4. Therefore we can see the improved performance of the model in each round:



```
PS C:\Users\Anvith\OneDrive\Desktop\Uni\SEM 7 LABS\DE LAB\experiments\MNIST\clients> python .\client1.py
2023-12-20 21:29:13.227257: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in pe
rformance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropr
iate compiler flags.
INFO flwr 2023-12-20 21:29:13,771 | grpc.py:49 | Opened insecure gRPC connection (no certificates were passed)
DEBUG flwr 2023-12-20 21:29:13,773 | connection.py:42 | ChannelConnectivity.IDLE
DEBUG flwr 2023-12-20 21:29:13,777 | connection.py:42 | ChannelConnectivity.CONNECTING
DEBUG flwr 2023-12-20 21:29:13,790 | connection.py:42 | ChannelConnectivity.READY

Fit History :
 {'loss': [0.20198310911655426], 'accuracy': [0.9420472383499146], 'val_loss': [2.791046380996704], 'val_accuracy': [0.5026999711990356]}


GLOBAL Model Evaluation accuracy :  0.8065999746322632


Fit History :
 {'loss': [0.10879632830619812], 'accuracy': [0.9683464765548706], 'val_loss': [2.136277675628662], 'val_accuracy': [0.5336999893188477]}


GLOBAL Model Evaluation accuracy :  0.8611999750137329


Fit History :
 {'loss': [0.07780256122350693], 'accuracy': [0.9764304757118225], 'val_loss': [1.2930419445037842], 'val_accuracy': [0.6746000051498413]}


GLOBAL Model Evaluation accuracy :  0.9126999974250793


Fit History :
 {'loss': [0.06096537411212921], 'accuracy': [0.9807348847389221], 'val_loss': [1.2706661224365234], 'val_accuracy': [0.6944000124931335]}


GLOBAL Model Evaluation accuracy :  0.928600013256073

DEBUG flwr 2023-12-20 21:32:05,580 | connection.py:139 | gRPC channel closed
INFO flwr 2023-12-20 21:32:05,580 | app.py:215 | Disconnect and shut down
PS C:\Users\Anvith\OneDrive\Desktop\Uni\SEM 7 LABS\DE LAB\experiments\MNIST\clients>
```

We can see that although using a non-IID dataset, we can achieve a better performance (compared to traditional machine learning) for a task when we are training the model in a Federated way. This is mainly because the server aggregates the model updates sent by the client and computes the global model parameters.

After the 4 rounds of federated learning, we can see that the accuracy of the model increased from **80%** to **92%**.

Here is the codebase for the above experiment conducted:
https://github.com/anvithh/Flower-Federated-Learning-on-MNIST.

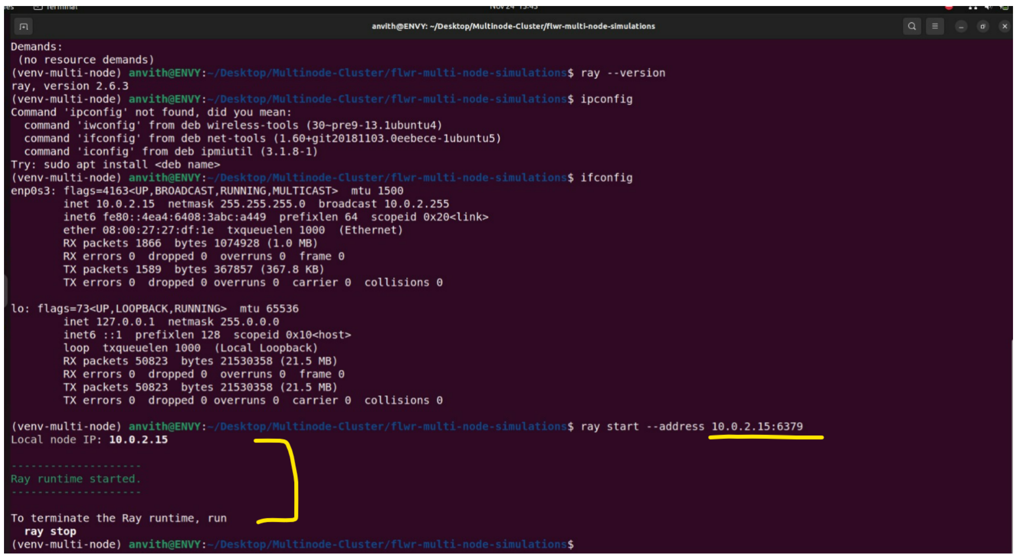
2.2 Implementing a FL Setup using multinode clusters

In the previous section, we explored one of the ways in which FLOWER allows us to implement FL setup *i.e.*, Simulation-based: multiple terminals depicting each client.

It also facilitates another way to achieve the same by using a **multi node setup**.

In this multi-node setup, virtual clients are generated using **Ray Actors**. This method proves to work better especially when we are dealing with a substantial no of clients with high computations required.

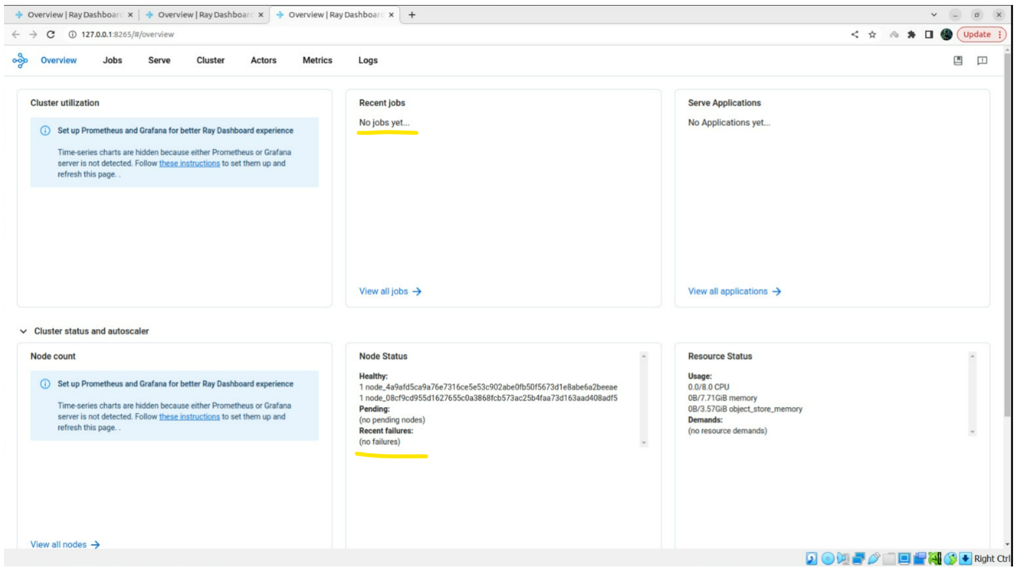
In this multi-node cluster, I designated one node (or PC) as the head node, and two nodes as clients.



```
anvith@ENVY: ~/Desktop/Multinode-Cluster/flwr-multi-node-simulations
Demands:
 (no resource demands)
(venv-multi-node) anvith@ENVY:~/Desktop/Multinode-Cluster/flwr-multi-node-simulations$ ray --version
ray, version 2.6.3
(venv-multi-node) anvith@ENVY:~/Desktop/Multinode-Cluster/flwr-multi-node-simulations$ ipconfig
Command 'ipconfig' not found, did you mean:
  command 'iwconfig' from deb wireless-tools (30~pre9-13.1ubuntu4)
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
  command 'iconfig' from deb ipmiutil (3.1.8-1)
Try: sudo apt install <deb name>
(venv-multi-node) anvith@ENVY:~/Desktop/Multinode-Cluster/flwr-multi-node-simulations$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::4ea4:6408:3abc:a449  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:27:df:1e  txqueuelen 1000  (Ethernet)
        RX packets 1866  bytes 1074928 (1.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1589  bytes 367857 (367.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 50823  bytes 21530358 (21.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 50823  bytes 21530358 (21.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(venv-multi-node) anvith@ENVY:~/Desktop/Multinode-Cluster/flwr-multi-node-simulations$ ray start --address 10.0.2.15:6379
Local node IP: 10.0.2.15

--------------------
Ray runtime started.
--------------------

To terminate the Ray runtime, run
  ray stop
(venv-multi-node) anvith@ENVY:~/Desktop/Multinode-Cluster/flwr-multi-node-simulations$
```

In the above picture, we can see that my laptop has started to act as the head node and it is accessible to other nodes at the Address: **10.2.0.15:6379**

Since no node has approached the head node **yet**, we can see in the below picture that no tasks are being done. (next page)



Cluster utilization

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page. .

Recent jobs

No jobs yet...

Serve Applications

No Applications yet...

Cluster status and autoscaler

Node count

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page. .

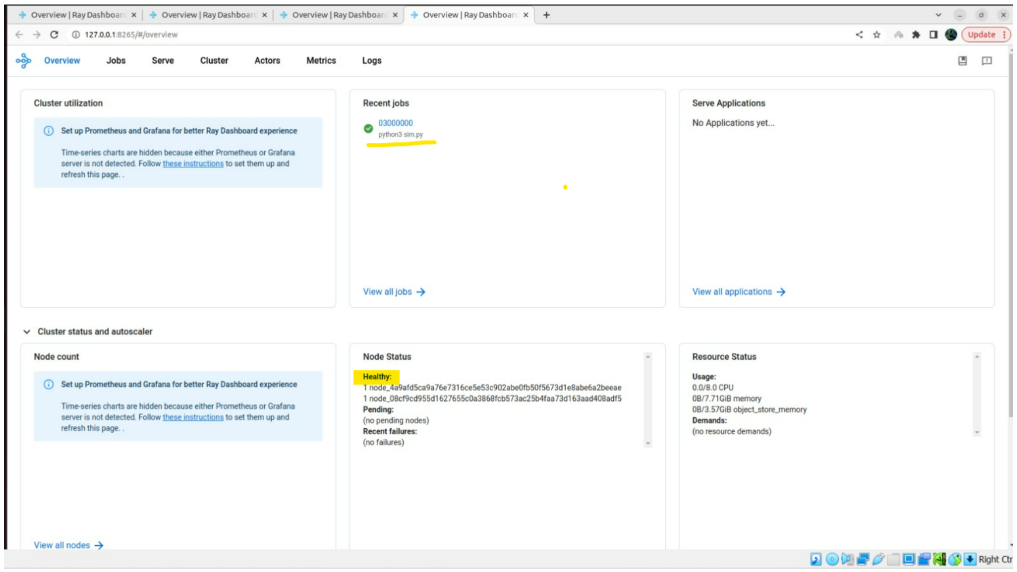
Node Status

```
Healthy:
1 node_4a9afd5ca9a76e7316ce5e53c902abe0fb50f5673d1e8abe6a2beeae
1 node_08cf9cd955d1627655c0a3868fcb573ac25b4faa73d163aad408adf5
Pending:
(no pending nodes)
Recent failures:
(no failures)
```

Resource Status

```
Usage:
0.0/8.0 CPU
0B/7.71GiB memory
0B/3.57GiB object_store_memory
Demands:
(no resource demands)
```

After a node has contacted the head node, the job execution begins and the same is reflected in the "Recent jobs" section in the below image. The python file ***sim.py*** has been successfully executed in the Federated Environment with **2** nodes.



Cluster utilization

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page. .

Recent jobs

03000000
python3 sim.py

Serve Applications

No Applications yet...

Cluster status and autoscaler

Node count

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page. .

Node Status

```
Healthy:
1 node_4a9afd5ca9a76e7316ce5e53c902abe0fb50f5673d1e8abe6a2beeae
1 node_08cf9cd955d1627655c0a3868fcb573ac25b4faa73d163aad408adf5
Pending:
(no pending nodes)
Recent failures:
(no failures)
```

Resource Status

```
Usage:
0.0/8.0 CPU
0B/7.71GiB memory
0B/3.57GiB object_store_memory
Demands:
(no resource demands)
```

2.3 Integrating Whisper Model in FL setup

I've integrated the Whisper-small model into the codebase and utilized a dataset that has been pre-processed by Vishnu for training.

2.3.1 Setup

The required libraries for federated learning, multiprocessing and speech processing have been imported. (flwr, multiprocessing, transformers etc)

2.3.2 Data Preparation

The audio data is preprocessed by resampling it (sampling_rate=16000) and extracting the log-Mel features.

Using the tokenizer, we encode the text labels into numerical IDs to input into the transformer.

A data collator is created to handle batching of audio and text data.

2.3.3 Federated Client

A flower class client is defined that participates in the Federated Learning.

The three functions of the federated client: *get_parameters*, *fit* and *evaluate* are written to perform their respective tasks.

The above methods are implemented to get the model parameters, train the model locally on the private dataset and evaluate the model on the same.

Finally, the client receives global model parameters, trains locally, and sends back updated parameters to the server for aggregation.