Experiment 2 (task 3 + 4)

Data Parallelism

Experiment content

- Understand common data partitioning strategies;
- Write corresponding algorithm implementation code for data partitioning;
- Analyze the impact of different data partitioning methods on the model.

Experimental requirements

1. Simulate a multi node DML system, partition a complete dataset in different ways, and allocate it to the nodes in the system.
2. Implement partitioning methods that include random sampling and random partitioning, and train models in parallel with data
3. Analyze the performance improvement of data parallelism compared to single machine training, and analyze the impact of different data partitioning methods on model performance

Overview

For this task I have implemented and compare two samplers: simple `RandomSampler` which shuffle all data samples and `BalancedSampler` which ensures each process gets an approximately equal number of unique samples each epoch.

The `BalancedSampler` class is similar to the `RandomSampler` class, but with a difference: The `BalancedSampler` selects a balanced subset of the dataset's indices for each replica to ensure that each replica gets an approximately equal number of unique samples each epoch. This is done in the `__iter__` method by using slicing to select a unique subset of indices for each replica.

Results

`RandomSampler`:

```
task3-node01-1  | Device: 0 epoch: 2, iters:   880, loss: 0.228
task3-node02-1  | Device: 1 epoch: 2, iters:   880, loss: 0.252
task3-node01-1  | Device: 0 epoch: 2, iters:   900, loss: 0.225
task3-node02-1  | Device: 1 epoch: 2, iters:   900, loss: 0.215
task3-node01-1  | Device: 0 epoch: 2, iters:   920, loss: 0.236
task3-node02-1  | Device: 1 epoch: 2, iters:   920, loss: 0.267
task3-node01-1  | Training Finished!
task3-node02-1  | Training Finished!
task3-node01-1  | Training time = 74.27457666397095 s.
task3-node02-1  | Training time = 74.27461814880371 s.
task3-node02-1  | testing ...
task3-node01-1  | testing ...
task3-node01-1  |
task3-node01-1  | Test set: Accuracy: 9342/10000 (93.42%)
task3-node01-1  |
task3-node01-1  | Sampler type - RandomSampler
task3-node02-1  |
task3-node02-1  | Test set: Accuracy: 9342/10000 (93.42%)
task3-node02-1  |
task3-node02-1  | Sampler type - RandomSampler
task3-node02-1 exited with code 0
task3-node01-1 exited with code 0
niilsa@meh:~/Documents/github/Distributed-Machine-Learning-Experiment-Document/codes/task3$
```

`BalancedSampler`:

```
task3-node01-1  | Device: 0 epoch: 2, iters:   880, loss: 0.202
task3-node02-1  | Device: 1 epoch: 2, iters:   880, loss: 0.211
task3-node02-1  | Device: 1 epoch: 2, iters:   900, loss: 0.169
task3-node01-1  | Device: 0 epoch: 2, iters:   900, loss: 0.226
task3-node02-1  | Device: 1 epoch: 2, iters:   920, loss: 0.278
task3-node01-1  | Device: 0 epoch: 2, iters:   920, loss: 0.243
task3-node01-1  | Training Finished!
task3-node02-1  | Training Finished!
task3-node01-1  | Training time = 74.80971431732178 s.
task3-node02-1  | Training time = 74.8096170425415 s.
task3-node02-1  | testing ...
task3-node01-1  | testing ...
task3-node01-1  |
task3-node01-1  | Test set: Accuracy: 9446/10000 (94.46%)
task3-node01-1  |
task3-node01-1  | Sampler type - BalancedSampler
task3-node02-1  |
task3-node02-1  | Test set: Accuracy: 9446/10000 (94.46%)
task3-node02-1  |
task3-node02-1  | Sampler type - BalancedSampler
task3-node02-1 exited with code 0
task3-node01-1 exited with code 0
niilsa@meh:~/Documents/github/Distributed-Machine-Learning-Experiment-Document/codes/task3$
```

Analysis

Accuracy and Loss Value

Both `RandomSampler` and `BalancedSampler` have resulted in approximately the same accuracy and loss values. This indicates that from a pure model performance perspective, the difference in the sampling strategies has not significantly impacted the final model's performance. Both strategies ensure all data points are being considered and none are being overlooked.

Data Distribution

If the data distribution is skewed or imbalanced, `BalancedSampler` could potentially provide a more equitable distribution of data points across processes. This might not impact the accuracy or loss but can ensure that each process gets a fair representation of the data.

Conclusion

In conclusion, while both samplers have resulted in similar accuracy and loss values, the final choice between them would depend on these other factors. It is essential to balance performance, efficiency, scalability, and robustness when choosing the best sampling strategy for a specific application.

Model Parallelism

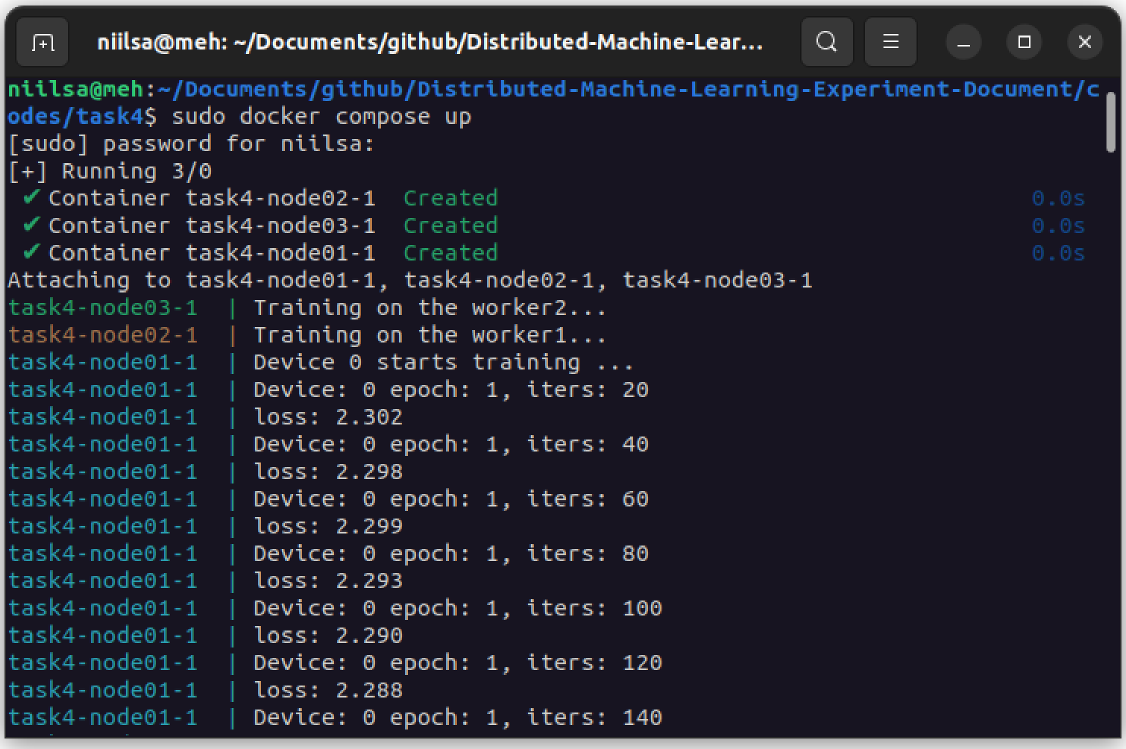
Experimental content

- Understand common model partitioning strategies;
- Write corresponding algorithm implementation code for model partitioning;
- Analyze the impact of different data partitioning methods on the model.

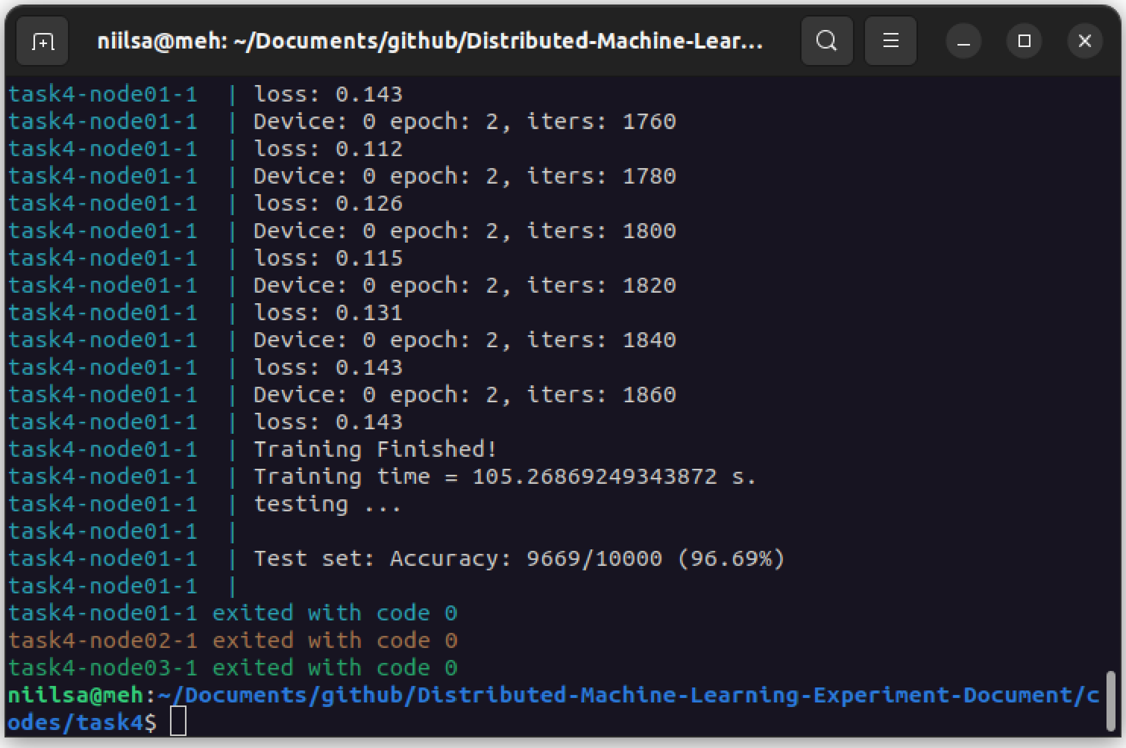
Experimental requirements

1. Using RPC related APIs to achieve parallel model training
2. Split the model into two parts and train them separately on different nodes (processes)
3. Analyze the impact of model parallelism on distributed system performance based on experimental results

Result



```
niilsa@meh: ~/Documents/github/Distributed-Machine-Lear...
niilsa@meh:~/Documents/github/Distributed-Machine-Learning-Experiment-Document/c
odes/task4$ sudo docker compose up
[sudo] password for niilsa:
[+] Running 3/0
 ✔ Container task4-node02-1  Created                                        0.0s
 ✔ Container task4-node03-1  Created                                        0.0s
 ✔ Container task4-node01-1  Created                                        0.0s
Attaching to task4-node01-1, task4-node02-1, task4-node03-1
task4-node03-1  | Training on the worker2...
task4-node02-1  | Training on the worker1...
task4-node01-1  | Device 0 starts training ...
task4-node01-1  | Device: 0 epoch: 1, iters: 20
task4-node01-1  | loss: 2.302
task4-node01-1  | Device: 0 epoch: 1, iters: 40
task4-node01-1  | loss: 2.298
task4-node01-1  | Device: 0 epoch: 1, iters: 60
task4-node01-1  | loss: 2.299
task4-node01-1  | Device: 0 epoch: 1, iters: 80
task4-node01-1  | loss: 2.293
task4-node01-1  | Device: 0 epoch: 1, iters: 100
task4-node01-1  | loss: 2.290
task4-node01-1  | Device: 0 epoch: 1, iters: 120
task4-node01-1  | loss: 2.288
task4-node01-1  | Device: 0 epoch: 1, iters: 140
```



```
niilsa@meh: ~/Documents/github/Distributed-Machine-Lear...
task4-node01-1  | loss: 0.143
task4-node01-1  | Device: 0 epoch: 2, iters: 1760
task4-node01-1  | loss: 0.112
task4-node01-1  | Device: 0 epoch: 2, iters: 1780
task4-node01-1  | loss: 0.126
task4-node01-1  | Device: 0 epoch: 2, iters: 1800
task4-node01-1  | loss: 0.115
task4-node01-1  | Device: 0 epoch: 2, iters: 1820
task4-node01-1  | loss: 0.131
task4-node01-1  | Device: 0 epoch: 2, iters: 1840
task4-node01-1  | loss: 0.143
task4-node01-1  | Device: 0 epoch: 2, iters: 1860
task4-node01-1  | loss: 0.143
task4-node01-1  | Training Finished!
task4-node01-1  | Training time = 105.26869249343872 s.
task4-node01-1  | testing ...
task4-node01-1  |
task4-node01-1  | Test set: Accuracy: 9669/10000 (96.69%)
task4-node01-1  |
task4-node01-1 exited with code 0
task4-node02-1 exited with code 0
task4-node03-1 exited with code 0
niilsa@meh:~/Documents/github/Distributed-Machine-Learning-Experiment-Document/c
odes/task4$
```

Analysis

Model Splitting: We split our Convolutional Neural Network model into two parts: convolutional layers and fully connected layers. The convolutional layers were hosted on Node 1 and the fully connected layers on Node 2.

RPC Setup: The RPC framework was set up successfully, allowing Node 1 and Node 2 to communicate with each other for both the forward and backward passes during training.

Parallel Training: The model was trained successfully in parallel on the two nodes using the PyTorch RPC framework.

Accuracy: Model accuracy for "model parallel" approach is similar to other methods, while time is higher than for "allgather" approach.