Parallel Machine Learning with Horovod

Luca Venerando Greco, Elia Vaglietti, Bice Marzagora

1 Introduction

The goal of this project is to implement distributed deep learning using Horovod to classify CIFAR-10 images and report performance improvements. We followed the official Horovod tutorial (*https://ulhpc-tutorials.readthedocs.io/en/latest/deep_learning/horovod/*). In addition to implementing the tutorial's sample code, we conducted strong scalability tests to evaluate the system's performance under increasing numbers of compute nodes, providing insights into the efficiency and speedup gains of the distributed approach.

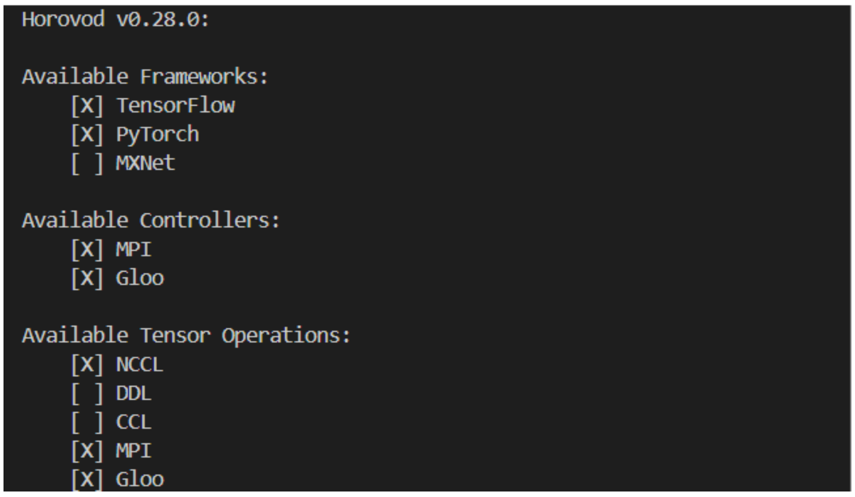
2 Steps Followed in the Tutorial

2.1 Environment Setup

The tutorial provides two approaches: using a pre-installed environment or setting it up manually. For simplicity and consistency, we chose the pre-installed approach.

```
source /work/projects/ulhpc-tutorials/PS10-Horovod/env.sh

horovodrun --check-build
```

Screenshot: Verification of Horovod setup.

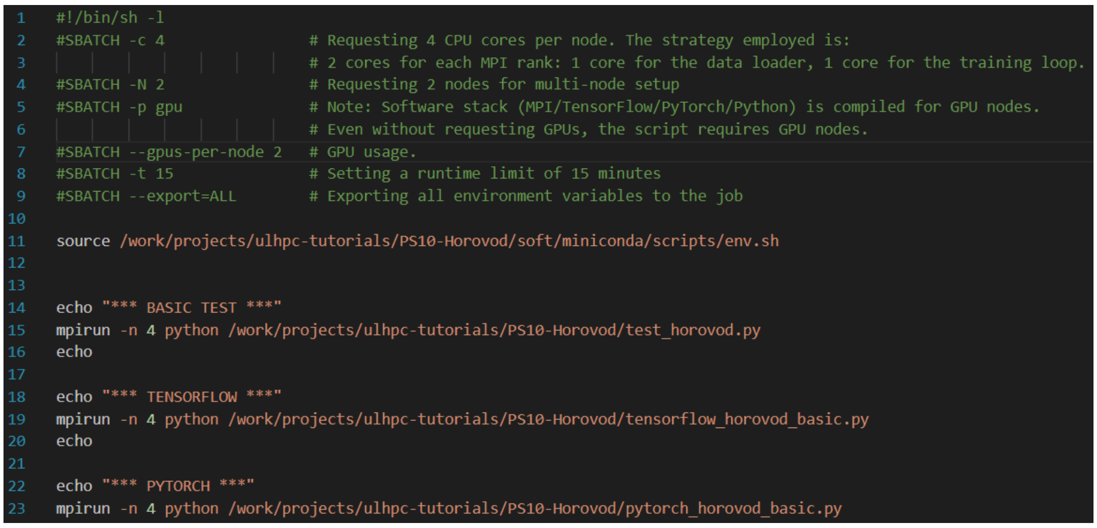


```
Horovod v0.28.0:

Available Frameworks:
    [X] TensorFlow
    [X] PyTorch
    [ ] MXNet

Available Controllers:
    [X] MPI
    [X] Gloo

Available Tensor Operations:
    [X] NCCL
    [ ] DDL
    [ ] CCL
    [X] MPI
    [X] Gloo
```

2.2 Code samples test

To validate the setup and explore Horovod's functionality, we executed the code samples provided in the tutorial. The following SLURM job script was used to configure the environment and execute the tests:

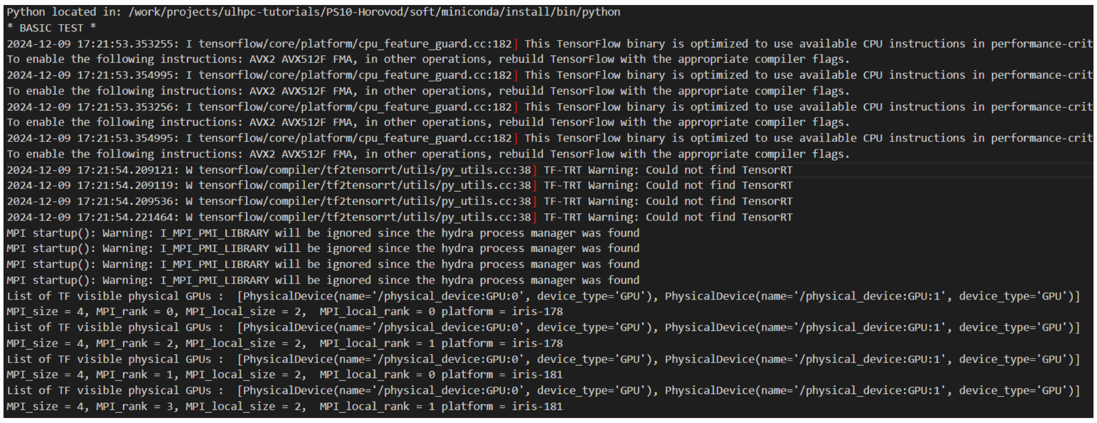
Screenshot: slurm job.



```
#!/bin/sh -l
#SBATCH -c 4                 # Requesting 4 CPU cores per node. The strategy employed is:
                             # 2 cores for each MPI rank: 1 core for the data loader, 1 core for the training loop.
#SBATCH -N 2                 # Requesting 2 nodes for multi-node setup
#SBATCH -p gpu               # Note: Software stack (MPI/TensorFlow/PyTorch/Python) is compiled for GPU nodes.
                             # Even without requesting GPUs, the script requires GPU nodes.
#SBATCH --gpus-per-node 2    # GPU usage.
#SBATCH -t 15                # Setting a runtime limit of 15 minutes
#SBATCH --export=ALL         # Exporting all environment variables to the job

source /work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/scripts/env.sh


echo "*** BASIC TEST ***"
mpirun -n 4 python /work/projects/ulhpc-tutorials/PS10-Horovod/test_horovod.py
echo

echo "*** TENSORFLOW ***"
mpirun -n 4 python /work/projects/ulhpc-tutorials/PS10-Horovod/tensorflow_horovod_basic.py
echo

echo "*** PYTORCH ***"
mpirun -n 4 python /work/projects/ulhpc-tutorials/PS10-Horovod/pytorch_horovod_basic.py
```

- Test code for Horovod initialization (test_horovod.py):

  Screenshot: Output of the job execution .

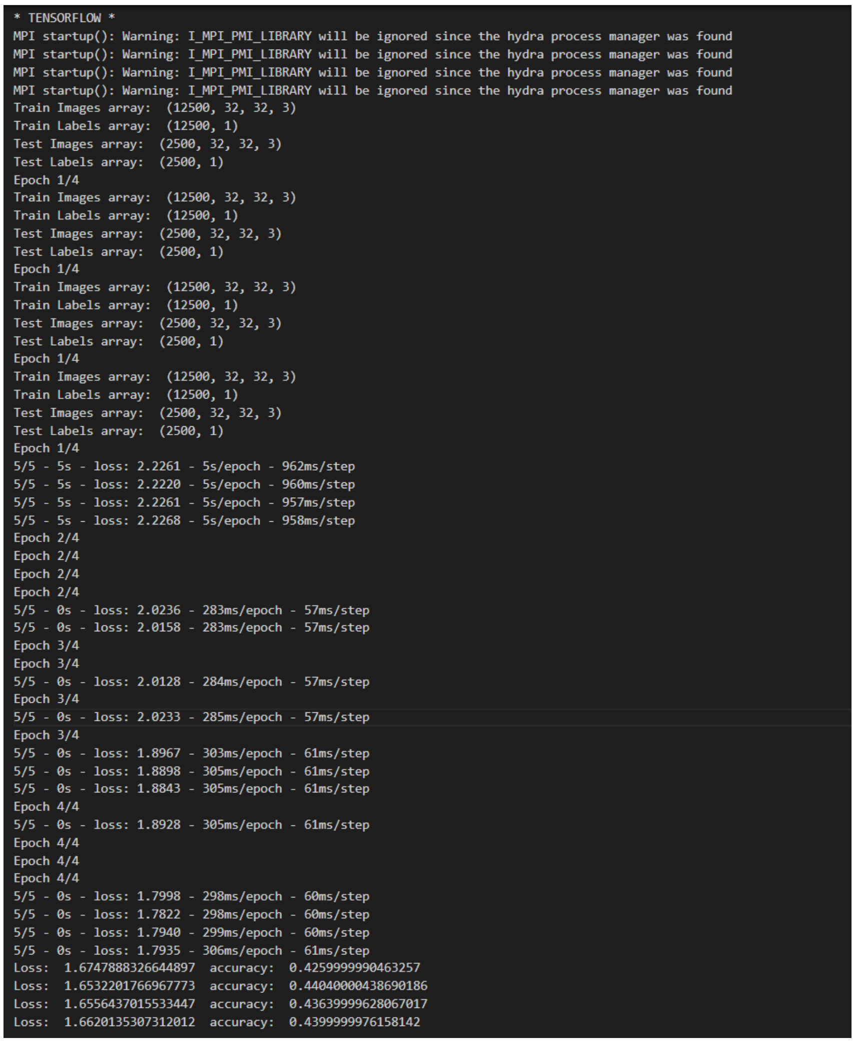


```
Python located in: /work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/install/bin/python
* BASIC TEST *
2024-12-09 17:21:53.353255: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-crit
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-09 17:21:53.354995: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-crit
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-09 17:21:53.353256: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-crit
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-09 17:21:53.354995: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-crit
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-09 17:21:54.209121: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-12-09 17:21:54.209119: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-12-09 17:21:54.209536: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-12-09 17:21:54.221464: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
List of TF visible physical GPUs :  [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
MPI_size = 4, MPI_rank = 0, MPI_local_size = 2,  MPI_local_rank = 0 platform = iris-178
List of TF visible physical GPUs :  [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
MPI_size = 4, MPI_rank = 2, MPI_local_size = 2,  MPI_local_rank = 1 platform = iris-178
List of TF visible physical GPUs :  [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
MPI_size = 4, MPI_rank = 1, MPI_local_size = 2,  MPI_local_rank = 0 platform = iris-181
List of TF visible physical GPUs :  [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
MPI_size = 4, MPI_rank = 3, MPI_local_size = 2,  MPI_local_rank = 1 platform = iris-181
```

  The basic test confirms successful execution. Four MPI ranks were distributed across two nodes, each with two GPUs, and all ranks identified their assigned GPUs correctly. TensorFlow issued warnings about missing AVX optimizations and TensorRT, but these do not affect training. MPI-related warnings were also observed but did not impact execution.

- ULHPC TensorFlow/Keras Example (tensorflow_horovod_basic.py)::

  Screenshot: Output of the job execution.



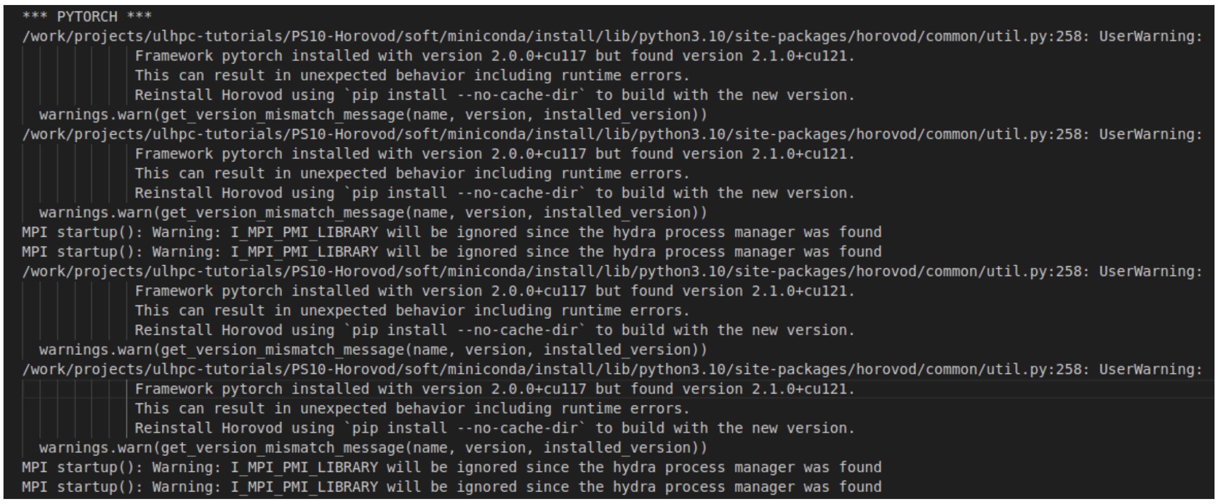
```
* TENSORFLOW *
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
Train Images array:  (12500, 32, 32, 3)
Train Labels array:  (12500, 1)
Test Images array:  (2500, 32, 32, 3)
Test Labels array:  (2500, 1)
Epoch 1/4
Train Images array:  (12500, 32, 32, 3)
Train Labels array:  (12500, 1)
Test Images array:  (2500, 32, 32, 3)
Test Labels array:  (2500, 1)
Epoch 1/4
Train Images array:  (12500, 32, 32, 3)
Train Labels array:  (12500, 1)
Test Images array:  (2500, 32, 32, 3)
Test Labels array:  (2500, 1)
Epoch 1/4
Train Images array:  (12500, 32, 32, 3)
Train Labels array:  (12500, 1)
Test Images array:  (2500, 32, 32, 3)
Test Labels array:  (2500, 1)
Epoch 1/4
5/5 - 5s - loss: 2.2261 - 5s/epoch - 962ms/step
5/5 - 5s - loss: 2.2220 - 5s/epoch - 960ms/step
5/5 - 5s - loss: 2.2261 - 5s/epoch - 957ms/step
5/5 - 5s - loss: 2.2268 - 5s/epoch - 958ms/step
Epoch 2/4
Epoch 2/4
Epoch 2/4
Epoch 2/4
5/5 - 0s - loss: 2.0236 - 283ms/epoch - 57ms/step
5/5 - 0s - loss: 2.0158 - 283ms/epoch - 57ms/step
Epoch 3/4
Epoch 3/4
5/5 - 0s - loss: 2.0128 - 284ms/epoch - 57ms/step
Epoch 3/4
5/5 - 0s - loss: 2.0233 - 285ms/epoch - 57ms/step
Epoch 3/4
5/5 - 0s - loss: 1.8967 - 303ms/epoch - 61ms/step
5/5 - 0s - loss: 1.8898 - 305ms/epoch - 61ms/step
5/5 - 0s - loss: 1.8843 - 305ms/epoch - 61ms/step
Epoch 4/4
5/5 - 0s - loss: 1.8928 - 305ms/epoch - 61ms/step
Epoch 4/4
Epoch 4/4
Epoch 4/4
5/5 - 0s - loss: 1.7998 - 298ms/epoch - 60ms/step
5/5 - 0s - loss: 1.7822 - 298ms/epoch - 60ms/step
5/5 - 0s - loss: 1.7940 - 299ms/epoch - 60ms/step
5/5 - 0s - loss: 1.7935 - 306ms/epoch - 61ms/step
Loss:  1.6747888326644897  accuracy:  0.4259999990463257
Loss:  1.6532201766967773  accuracy:  0.44040000438690186
Loss:  1.6556437015533447  accuracy:  0.43639999628067017
Loss:  1.6620135307312012  accuracy:  0.4399999976158142
```

This output confirms successful distributed training of a TensorFlow model using Horovod with 4 MPI ranks. The training data (12,500 images) and testing data (2,500 images) were correctly loaded, and loss values steadily decreased over 4 epochs.

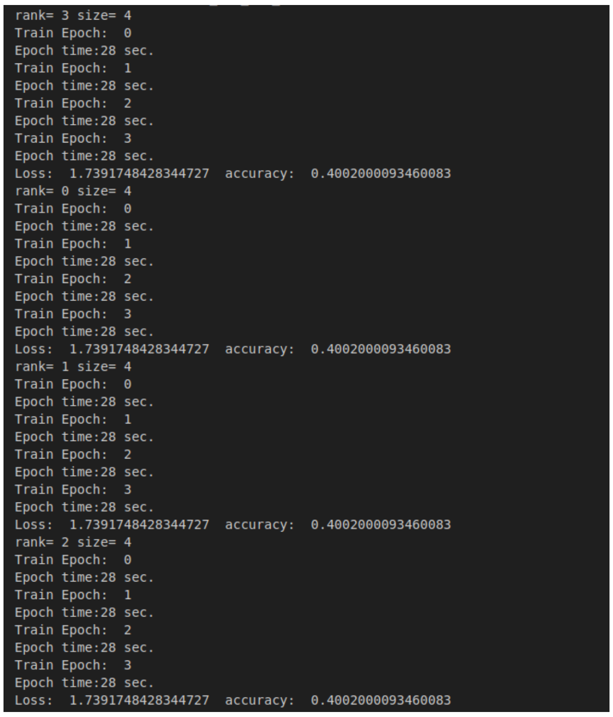
- ULHPC PyTorch Example:

  We ran these tests without GPUs beacuse running the same tests with GPUs resulted in segmentation faults and warnings about a version mismatch between PyTorch ('2.1.0+cu121'), Horovod (built for '2.0.0+cu117'), and the CUDA/cuDNN stack. Running on the CPU backend bypasses these issues,because the problem is specific to the GPU setup and requires version alignment for successful execution.

  Screenshot: Training execution example using ULHPC pythorch code, using no GPUs.

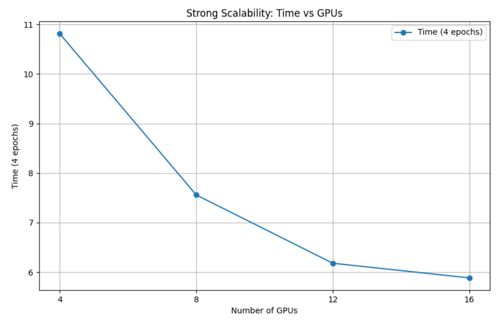


```
*** PYTORCH ***
/work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/install/lib/python3.10/site-packages/horovod/common/util.py:258: UserWarning:
             Framework pytorch installed with version 2.0.0+cu117 but found version 2.1.0+cu121.
             This can result in unexpected behavior including runtime errors.
             Reinstall Horovod using `pip install --no-cache-dir` to build with the new version.
  warnings.warn(get_version_mismatch_message(name, version, installed_version))
/work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/install/lib/python3.10/site-packages/horovod/common/util.py:258: UserWarning:
             Framework pytorch installed with version 2.0.0+cu117 but found version 2.1.0+cu121.
             This can result in unexpected behavior including runtime errors.
             Reinstall Horovod using `pip install --no-cache-dir` to build with the new version.
  warnings.warn(get_version_mismatch_message(name, version, installed_version))
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
/work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/install/lib/python3.10/site-packages/horovod/common/util.py:258: UserWarning:
             Framework pytorch installed with version 2.0.0+cu117 but found version 2.1.0+cu121.
             This can result in unexpected behavior including runtime errors.
             Reinstall Horovod using `pip install --no-cache-dir` to build with the new version.
  warnings.warn(get_version_mismatch_message(name, version, installed_version))
/work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/install/lib/python3.10/site-packages/horovod/common/util.py:258: UserWarning:
             Framework pytorch installed with version 2.0.0+cu117 but found version 2.1.0+cu121.
             This can result in unexpected behavior including runtime errors.
             Reinstall Horovod using `pip install --no-cache-dir` to build with the new version.
  warnings.warn(get_version_mismatch_message(name, version, installed_version))
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
```

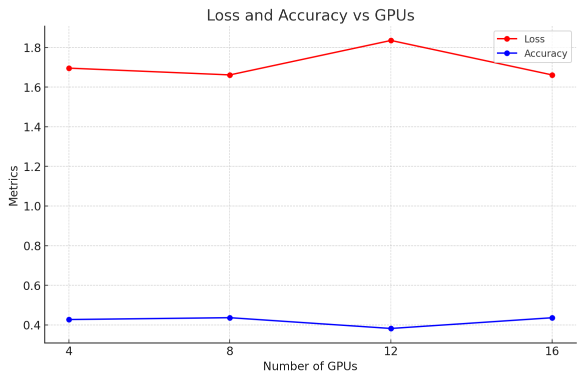


```
rank= 3 size= 4
Train Epoch:  0
Epoch time:28 sec.
Train Epoch:  1
Epoch time:28 sec.
Train Epoch:  2
Epoch time:28 sec.
Train Epoch:  3
Epoch time:28 sec.
Loss:  1.7391748428344727  accuracy:  0.4002000093460083
rank= 0 size= 4
Train Epoch:  0
Epoch time:28 sec.
Train Epoch:  1
Epoch time:28 sec.
Train Epoch:  2
Epoch time:28 sec.
Train Epoch:  3
Epoch time:28 sec.
Loss:  1.7391748428344727  accuracy:  0.4002000093460083
rank= 1 size= 4
Train Epoch:  0
Epoch time:28 sec.
Train Epoch:  1
Epoch time:28 sec.
Train Epoch:  2
Epoch time:28 sec.
Train Epoch:  3
Epoch time:28 sec.
Loss:  1.7391748428344727  accuracy:  0.4002000093460083
rank= 2 size= 4
Train Epoch:  0
Epoch time:28 sec.
Train Epoch:  1
Epoch time:28 sec.
Train Epoch:  2
Epoch time:28 sec.
Train Epoch:  3
Epoch time:28 sec.
Loss:  1.7391748428344727  accuracy:  0.4002000093460083
```

2.3 Testing strong scalability



(a) Strong Scalability: Time vs GPUs

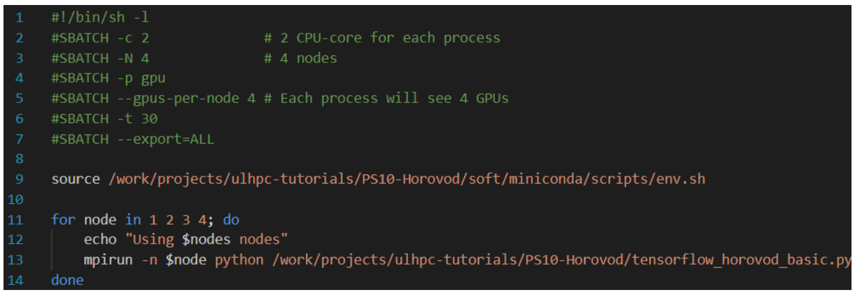


(b) Loss and Accuracy vs GPUs

Figure 1: Strong scalability and performance metrics as the number of GPUs increases.

In general, as the number of nodes increases, the computation becomes more distributed, resulting in faster training times due to the parallelization of workloads. The first plot shows a decrease in training time as the number of GPUs increases. The second plot highlights model performance metrics, loss and accuracy, which remain constant as expected.

The SLURM job script used to run this test is as follows:



```
#!/bin/sh -l
#SBATCH -c 2              # 2 CPU-core for each process
#SBATCH -N 4              # 4 nodes
#SBATCH -p gpu
#SBATCH --gpus-per-node 4 # Each process will see 4 GPUs
#SBATCH -t 30
#SBATCH --export=ALL

source /work/projects/ulhpc-tutorials/PS10-Horovod/soft/miniconda/scripts/env.sh

for node in 1 2 3 4; do
    echo "Using $nodes nodes"
    mpirun -n $node python /work/projects/ulhpc-tutorials/PS10-Horovod/tensorflow_horovod_basic.py
done
```