MES Wadia College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-05**

Title of the Assignment: Implement HPC application for AI/ML domain.

Objective: Students get hands-on experience in developing high-performance computing applications for the AI/ML domain. By completing this assignment, students will gain practical skills in AI/ML algorithms and models, programming languages, hardware architectures, data pre-processing and management, HPC system administration, and optimization and tuning.

Prerequisite:

1. **Knowledge of AI/ML algorithms and models:** knowledge of statistical methods, linear algebra, optimization techniques, and deep learning frameworks such as TensorFlow, PyTorch, and MXNet.
2. **Proficiency in programming languages:** such as C++, Python, and CUDA is essential to develop an HPC application for AI/ML.
3. **Knowledge of hardware architectures :** hardware architectures, such as CPU, GPU, FPGA, and ASIC

Theory:

High-performance computing (HPC) is a critical component of many AI/ML applications, particularly those that require large-scale training and inference on massive datasets. In this section, we will outline a general approach for implementing an HPC application for the AI/ML domain.

Problem Formulation: The first step in implementing an HPC application for AI/ML is to formulate the problem as a set of mathematical and computational tasks that can be parallelized and optimized.

This involves defining the problem domain, selecting appropriate algorithms and models, and determining the computational and memory requirements.

Hardware Selection: The next step is to select the appropriate hardware platform for the HPC application. This involves considering the available hardware options, such as CPU, GPU, FPGA, and ASIC, and selecting the most suitable option based on the performance, cost, power consumption, and scalability requirements.

Software Framework Selection: Once the hardware platform has been selected, the next step is to choose the appropriate software framework for the AI/ML application. This involves considering

the available options, such as TensorFlow, PyTorch, MXNet, and Caffe, and selecting the most suitable framework based on the programming language, performance, ease of use, and community support.

Data Preparation and Preprocessing: Before training or inference can be performed, the data must be prepared and pre-processed. This involves cleaning the data, normalizing and scaling the data, and splitting the data into training, validation, and testing sets. The data must also be stored in a format that is compatible with the selected software framework.

Model Training or Inference: The main computational task in an AI/ML application is model training or inference. In an HPC application, this task is parallelized and optimized to take advantage of the available hardware resources. This involves breaking the model into smaller tasks that can be parallelized, using techniques such as data parallelism, model parallelism, or pipeline parallelism. The performance of the application is optimized by reducing the communication overhead between nodes or GPUs, balancing the workload among nodes, and optimizing the memory access patterns.

Model Evaluation: After the model has been trained or inference has been performed, the performance of the model must be evaluated. This involves computing the accuracy, precision, recall, and other metrics on the validation and testing sets. The performance of the HPC application is evaluated by measuring the speedup, scalability, and efficiency of the parallelized task

Optimization and Tuning: Finally, the HPC application must be optimized and tuned to achieve the best possible performance. This involves profiling the code to identify bottlenecks and optimizing the code using techniques such as loop unrolling, vectorization, and cache optimization. The performance of the application is also affected by the choice of hyper parameters, such as the learning rate, batch size, and regularization strength, which must be tuned using techniques such as grid search or Bayesian optimization.

Application: Neural Network Training

Objective: Train a simple neural network on a large dataset of images using TensorFlow and HPC.

Approach: We will use TensorFlow to define and train the neural network and use a parallel computing framework to distribute the computation across multiple nodes in a cluster.

Requirements:

TensorFlow 2.0 or higher mpi4py

Steps:

Define the neural network architecture

Code:

```
import tensorflow as tf
model = tf.keras.models.Sequential([
tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),
```

```
tf.keras.layers.MaxPooling2D((2, 2)),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(10, activation='softmax')
])
```

Load the dataset:

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Initialize MPI

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
```

Define the training function:

```
def train(model, x_train, y_train, rank, size):
# Split the data across the nodes n =
len(x_train)
chunk_size = n // size start = rank *
chunk_size end = (rank + 1) * chunk_size
if rank == size - 1:
end = n
x_train_chunk = x_train[start:end]
y_train_chunk = y_train[start:end]
# Compile the model
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
# Train the model
model.fit(x_train_chunk, y_train_chunk, epochs=1, batch_size=32)
# Compute the accuracy on the training data
train_loss, train_acc = model.evaluate(x_train_chunk, y_train_chunk, verbose=2)
# Reduce the accuracy across all nodes
train_acc = comm.allreduce(train_acc, op=MPI.SUM)
return train_acc / size
```

Run the training loop:

```
epochs = 5
for epoch in range(epochs):
```

```
# Train the model
train_acc = train(model, x_train, y_train, rank, size)
# Compute the accuracy on the test data
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
# Reduce the accuracy across all nodes
test_acc = comm.allreduce(test_acc, op=MPI.SUM)
# Print the results if rank ==
0:
print(f"Epoch {epoch + 1}: Train accuracy = {train_acc:.4f}, Test accuracy = {test_acc /
size:.4f}")
```

Output:

Epoch 1: Train accuracy = 0.9773, Test accuracy = 0.9745

Epoch 2: Train accuracy = 0.9859, Test accuracy = 0.9835

Epoch 3: Train accuracy = 0.9887, Test accuracy = 0.9857

Epoch 4: Train accuracy = 0.9905, Test accuracy = 0.9876

Epoch 5: Train accuracy = 0.9919, Test accuracy = 0.9880

Conclusion: Implementing an HPC application for the AI/ML domain involves formulating the problem, selecting the hardware and software frameworks, preparing and pre-processing the data parallelizing and optimizing the model training or inference tasks, evaluating the model performance, and optimizing and tuning the HPC application for maximum performance. This requires expertise in mathematics, computer science, and domain-specific knowledge of AI/ML algorithms and models.

Question:

1. Explain how HPC is used in AI/ML?
2. Write note on TensorFlow, PyTorch, and MXNet.
3. State different application of AI/ML for HPC.
4. State advantage and disadvantage of using HPC in AI/ML