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 preprocessing 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.