

Bahria University
Department of Computer Science



Parallel and Distributive Computing

Group Members:

Ahmed Raza(01-134222-015)

Muhammad Arham(01-134231-102)

Abdul Jabbar(01-134221-002)

Shams Ul Islam (01-134212-166)

Syed Muhammad Ali Hassan (01-134212-176)

Task 2:

Consider two problems of classification/Regression. Acquire Datasets from Kaggle of Sentiment Analysis and Fraud Detection (Credit Card).

- i. Apply Data parallelism by sending data to the GPU using CUDA.
- ii. Apply Task Parallelism by using multi-threading.

Libraries to be used. Pandas, numpy, tensorflow

Solution:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import threading
```

```
# IMDb dataset (50,000 reviews, labeled positive/negative)
(x_train, y_train), (x_test, y_test) =
keras.datasets.imdb.load_data(num_words=10000)

# Pad sequences for equal length
x_train = keras.preprocessing.sequence.pad_sequences(x_train,
maxlen=200)
x_test = keras.preprocessing.sequence.pad_sequences(x_test, maxlen=200)
```

Output:

```
✓ Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ————— 2s 0us/step
```

```
def create_model():
    model = keras.Sequential([
        layers.Embedding(input_dim=10000, output_dim=64,
input_length=200),
        layers.LSTM(64),
        layers.Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
    return model
```

```
#Part I - Data Parallelism (CUDA)
with tf.device('/GPU:0'): # Forces training on GPU
    model = create_model()
    history = model.fit(x_train, y_train, batch_size=512, epochs=2,
validation_split=0.2)
```

Output:

```
Epoch 1/2
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
warnings.warn(
40/40 ————— 6s 32ms/step - accuracy: 0.5838 - loss: 0.6709 - val_accuracy: 0.7876 - val_loss: 0.4598
Epoch 2/2
40/40 ————— 1s 23ms/step - accuracy: 0.8366 - loss: 0.3834 - val_accuracy: 0.8334 - val_loss: 0.3801
```

```
#Part II - Task Parallelism (Multi-threading)
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import pandas as pd

def train_sentiment_model():
```

```

with tf.device('/GPU:0'):
    model = create_model()
    model.fit(x_train, y_train, batch_size=512, epochs=2,
validation_split=0.2)
    print("Sentiment model training completed")
#####3
def load_credit_approval_dataset():
    data = fetch_openml("credit-g", version=1, as_frame=True)
    df = data.frame
    return df

def fraud_preprocessing_task():
    # Load dataset
    df = load_credit_approval_dataset()
    print("Original dataset shape:", df.shape)
    print("Class distribution:\n", df['class'].value_counts(), "\n")

    # Separate features and target
    X = df.drop("class", axis=1)
    y = (df["class"] == "bad").astype(int) # encode: bad=1 (fraud),
good=0

    # Identify categorical vs numeric
    categorical_cols = X.select_dtypes(include=["category",
"object"]).columns
    numeric_cols = X.select_dtypes(exclude=["category",
"object"]).columns

    # Preprocessing: OneHot for categorical, StandardScaler for numeric
    preprocessor = ColumnTransformer(
        transformers=[
            ("num", StandardScaler(), numeric_cols),
            ("cat", OneHotEncoder(handle_unknown="ignore"),
categorical_cols)
        ]
    )

    # Pipeline
    pipeline = Pipeline(steps=[("preprocessor", preprocessor)])

```

```

X_processed = pipeline.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_processed, y, test_size=0.2, stratify=y, random_state=42
)

print("After preprocessing:")
print("Train set:", X_train.shape, " Test set:", X_test.shape)
print("Fraud cases in train:", sum(y_train), " / in test:",
sum(y_test))

    return X_train, X_test, y_train, y_test
#####3
# Run tasks in parallel using threads
t1 = threading.Thread(target=train_sentiment_model)
t2 = threading.Thread(target=fraud_preprocessing_task)

t1.start()
t2.start()

t1.join()
t2.join()

```

Output:

```

Original dataset shape: (1000, 21)
Class distribution:
class
good    700
bad     300
Name: count, dtype: int64

After preprocessing:
Train set: (800, 61) Test set: (200, 61)
Fraud cases in train: 240 / in test: 60
Epoch 1/2
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
warnings.warn(
40/40 ————— 2s 30ms/step - accuracy: 0.5865 - loss: 0.6618 - val_accuracy: 0.7986 - val_loss: 0.4370
Epoch 2/2
40/40 ————— 1s 22ms/step - accuracy: 0.8396 - loss: 0.3690 - val_accuracy: 0.8438 - val_loss: 0.3615
Sentiment model training completed

```

```
import threading
```

```

import time
import matplotlib.pyplot as plt

# --- Your Tasks ---
def train_sentiment_model():
    start = time.time()
    with tf.device('/GPU:0'):
        model = create_model()
        model.fit(x_train, y_train, batch_size=512, epochs=2,
validation_split=0.2)
        print("Sentiment model training completed")
    end = time.time()
    task_times["Sentiment Model"] = (start, end)

def load_credit_approval_dataset():
    data = fetch_openml("credit-g", version=1, as_frame=True)
    df = data.frame
    return df

def fraud_preprocessing_task():
    start = time.time()
    # Load dataset
    df = load_credit_approval_dataset()
    print("Original dataset shape:", df.shape)
    print("Class distribution:\n", df['class'].value_counts(), "\n")

    # Separate features and target
    X = df.drop("class", axis=1)
    y = (df["class"] == "bad").astype(int) # encode: bad=1 (fraud),
good=0

    # Identify categorical vs numeric
    categorical_cols = X.select_dtypes(include=["category",
"object"]).columns
    numeric_cols = X.select_dtypes(exclude=["category",
"object"]).columns

    # Preprocessing: OneHot for categorical, StandardScaler for numeric
    preprocessor = ColumnTransformer(

```

```

        transformers=[
            ("num", StandardScaler(), numeric_cols),
            ("cat", OneHotEncoder(handle_unknown="ignore"),
categorical_cols)
        ]
    )

    # Pipeline
    pipeline = Pipeline(steps=[("preprocessor", preprocessor)])
    X_processed = pipeline.fit_transform(X)

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(
        X_processed, y, test_size=0.2, stratify=y, random_state=42
    )

    print("After preprocessing:")
    print("Train set:", X_train.shape, " Test set:", X_test.shape)
    print("Fraud cases in train:", sum(y_train), " / in test:",
sum(y_test))
    end = time.time()
    task_times["Fraud Preprocessing"] = (start, end)

    return X_train, X_test, y_train, y_test

# Dictionary to store execution times
task_times = {}

# Run threads
t1 = threading.Thread(target=train_sentiment_model)
t2 = threading.Thread(target=fraud_preprocessing_task)

t1.start()
t2.start()
t1.join()
t2.join()

# --- Plot Gantt Chart ---

```

```

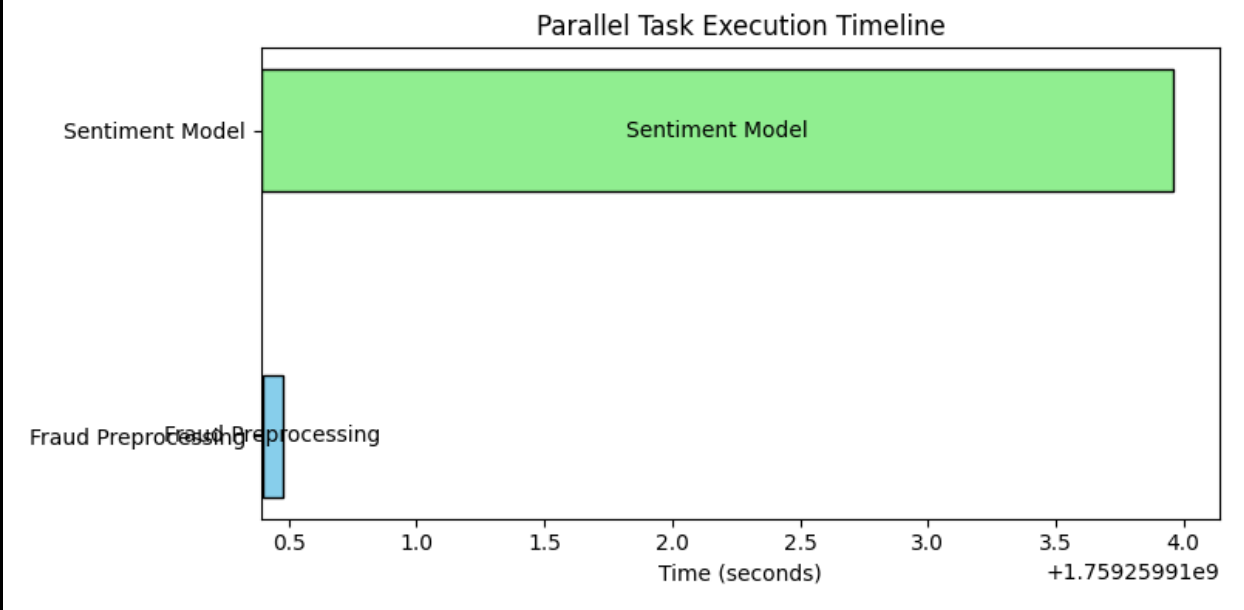
fig, ax = plt.subplots(figsize=(8,4))

colors = ["skyblue", "lightgreen"]
for i, (task, (start, end)) in enumerate(task_times.items()):
    ax.barh(y=i, width=end-start, left=start, height=0.4,
color=colors[i], edgecolor="black")
    ax.text((start+end)/2, i, task, ha="center", va="center",
fontSize=10, color="black")

ax.set_yticks([0, 1])
ax.set_yticklabels(list(task_times.keys()))
ax.set_xlabel("Time (seconds)")
ax.set_title("Parallel Task Execution Timeline")
plt.tight_layout()
plt.show()

```

Output:



Explanation:

Parallel Execution of Sentiment Model Training and Fraud Preprocessing

Thread 1: Sentiment Model Training

- Device: GPU
- Task: Train an LSTM neural network on IMDb movie reviews
- Data: 50,000 text reviews (positive/negative sentiment)
- Process:
 - Text preprocessing with an Embedding layer
 - LSTM sequence processing
 - Binary classification (sigmoid output)
- Duration: Longer running (neural network training)

Thread 2: Fraud Preprocessing

- Device: CPU
- Task: Preprocess credit approval dataset for fraud detection
- Data: Credit-g dataset with loan application data
- Process:
 - Load and clean financial data
 - Feature engineering (scaling numeric, encoding categorical)
 - Train-test split for fraud classification
- Duration: Shorter running (data preprocessing)

Parallel Execution Benefit

Both tasks run simultaneously:

- GPU handles compute-intensive neural network training
- CPU handles data loading and preprocessing
- Result: Reduced total execution time compared to sequential processing

The Gantt chart visualizes this overlap, showing how threading enables concurrent execution of different task types on appropriate hardware.

Github Link:

https://github.com/AhmedRaza15-dev/PDC_Assignment2

