**Parallelized Payment Fraud Detection System**

Hemant Gaikwad

CSYE7105 HIGH PARALLEL MACHINE LEARNING & AI – Prof. Handan Liu

**1. Introduction**

**1.1 Background**

With the rapid growth of e-commerce, online transactions have surged, leading to a corresponding rise in fraudulent activities. Online payment fraud results in significant financial losses for businesses and consumers, making it a critical issue to address. Recent advancements in machine learning and data analytics provide powerful tools for detecting and preventing fraud in real-time. Traditional fraud detection methods are often slow and ineffective against sophisticated fraud schemes, necessitating the development of high-performance solutions.

**1.2 Motivation**

The primary motivation is to enhance the security of online payment systems, protecting both consumers and businesses from fraud. There is a strong need for systems that can detect fraudulent transactions in real-time, minimizing losses and improving trust in online payment systems. The availability of large datasets, such as the one from Kaggle, allows for the training of robust machine learning models that can identify patterns indicative of fraud. Utilizing high-performance parallel computing can significantly speed up the training and inference processes, making it feasible to handle large volumes of transactions efficiently.

**1.3 Goal**

The primary goal is to develop a machine learning model that accurately classifies transactions as fraudulent or non-fraudulent using the provided dataset. Utilize parallel processing techniques to enhance the performance of the model, ensuring it can handle real-time transaction data. Assess the model's effectiveness using metrics such as accuracy, precision, recall, and F1-score to ensure it meets industry standards for fraud detection. Design a scalable solution that can be integrated into existing payment systems, providing ongoing protection against fraud as transaction volumes grow. Contribute to the field of online fraud detection by sharing findings and methodologies, potentially leading to further advancements in the area.

**Methodology**

1. Data Preparation

Data Extraction: Extract relevant features from the Kaggle online payment fraud detection dataset, focusing on transaction attributes such as transaction amount, user ID, timestamp, and device information.

Dataset Splitting: Split the dataset into training and testing subsets, ensuring that both sets maintain a balanced representation of fraudulent and non-fraudulent transactions to facilitate effective model training and evaluation.

2. Data Preprocessing

Data Loading: Utilize Distributed Data Parallel (DDP) to load the dataset across multiple GPUs or nodes, ensuring efficient memory utilization and parallel processing. Convert timestamps to datetime objects while maintaining consistency across distributed processes.

Feature Engineering: Preprocess the data by normalizing numerical features (e.g., scaling transaction amounts), encoding categorical variables (e.g., one-hot encoding for user IDs), and creating new features that may enhance model performance, such as transaction frequency or time since the last transaction.

Handling Imbalance: Apply resampling techniques, such as SMOTE (Synthetic Minority Over-sampling Technique) or random undersampling, to address class imbalance in the dataset, ensuring that the model is trained effectively on both classes.

3. Serial Implementation

Model Development: Develop a serial implementation to train and evaluate various machine learning models, including logistic regression, decision trees, random forests, and XGBoost, using a framework like Scikit-learn.

Baseline Evaluation: Establish baseline performance metrics for each model, including accuracy, precision, recall, and F1-score, to assess their effectiveness in detecting fraudulent transactions.

4. Parallelization

Identify Bottlenecks: Analyze the code to identify computationally intensive sections, particularly during data preprocessing and model training phases.

Implement Parallel Processing: Utilize parallel computing libraries such as Dask to distribute data processing and model training tasks across multiple CPUs and GPUs. This will enhance execution efficiency and reduce the time required for model training and evaluation.

5. Performance Comparison

Timing Analysis: Use timing functions to measure the execution time of each task in both the serial and parallel implementations, including data preprocessing, model training, and evaluation.

Execution Time Calculation: Record and compare the execution times for each stage, identifying specific areas where parallelization leads to significant performance improvements.

Metric Evaluation: Analyze the performance metrics of the models in both implementations to determine if parallelization affects model accuracy and efficiency.

6. Analysis and Visualization

Efficiency Analysis: Examine the differences in execution times between the serial and parallel implementations, highlighting the improvements achieved through parallel processing.

Visualization: Create visual representations, such as bar charts or line graphs, to clearly illustrate the performance results of the models and the time savings realized through parallelization.

**Data Description**

This dataset, with a size of 493.53 MB, contains historical records of online payment transactions, specifically focusing on fraudulent activities. Here are the key details:

Data Size: The dataset consists of 6,362,620 rows and 11 columns.

Data Types: int64: 3 columns, float64: 5 columns, object: 3 columns

Target Variable: isFraud: Indicates whether a transaction is fraudulent (1) or not (0).

**Data Source**

https://www.kaggle.com/datasets/jainilcoder/online-payment-fraud-detection