USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

BACHELOR OF SCIENCE (COMPUTER SCIENCE/SOFTWARE ENGINEERING)

CS428 Parallel and Distributed Computing

Efficient Data Processing Using Parallel and Distributed Processing

GROUP MEMBERS:

Muhammad Shahzaman 21B-207-CS
Salman Shahid 21B-185-CS
Shayan Naushad 21B-047 -CS
Abdul Wahab 21B-156-CS

Overview:

The project leverages Parallel and Distributed Processing (PDP) techniques to handle large datasets more efficiently, specifically focusing on the NYC Taxi Trip Dataset. The goal is to demonstrate a substantial reduction in processing time by implementing PDP, compared to traditional sequential data analysis methods.

Problem Statement:

Processing large datasets sequentially using traditional methods can be slow and inefficient, especially when working with big data. Sequential processing, which involves handling data one row at a time or in small chunks, often leads to long execution times for large datasets. Parallel and Distributed Processing (PDP) techniques, on the other hand, enable concurrent execution of tasks, significantly speeding up the data analysis process. This is particularly important for big data applications, where datasets can be so large that traditional methods become impractical.

Objectives:

1. **Performance Comparison**: Measure the difference in processing time between sequential processing (using pandas) and parallel processing (using Dask) for the NYC taxi data.
2. **OOP-based Code Structure**: Design the solution using Object-Oriented Programming (OOP) principles to ensure modularity, reusability, and scalability of the code.
3. **Time Reduction**: Demonstrate that using PDP techniques reduces the processing time by at least 60% when compared to sequential processing.

Methodology:

1. **Data Preparation**:
    - Load and clean the dataset (`yellow_tripdata_2015-01.csv`), ensuring that it is ready for analysis.
2. **Sequential Processing**:
    - Implement sequential data analysis using pandas. This will serve as the baseline for performance comparison.
    - Perform typical data processing tasks like filtering, grouping, and aggregating to understand the time it takes for pandas to process the dataset.
3. **Parallel Processing**:
    - Use Dask, a parallel computing library in Python, to implement parallel processing. Dask will be utilized to break the dataset into smaller chunks and process them concurrently.
    - Dask's parallelism will speed up tasks such as groupby operations, joins, and aggregations, which can be computationally expensive with large datasets.
4. **Performance Comparison**:
    - Measure and compare the execution time of both sequential (pandas) and parallel (Dask) methods using Python's `time` library.

- Ensure that the parallel processing time is at least 60% faster than the sequential approach.

Tools & Technologies:

- **Programming Language**: Python
- **Libraries**:
  - **pandas**: For data manipulation and analysis in sequential processing.
  - **Dask**: For distributed and parallel computing, helping to speed up data processing tasks.
  - **time**: To measure the execution time for both sequential and parallel processing.
- **Dataset**:
  - **NYC Taxi Trip Data** (`yellow_tripdata_2015-01.csv`): A dataset that contains detailed information about taxi trips in New York City.

Evaluation Criteria:

1. **Efficiency**: PDP (Dask) must complete the data processing in less than 2 minutes, which is 60% faster than the sequential method (which should take more than 5 minutes).
2. **Code Quality**: Ensure that the code follows object-oriented principles, has clear documentation, and is modular.
3. **Execution Time**: Sequential processing must take more than 5 minutes, while PDP should process the data in less than 2 minutes.
4. **Object-Oriented Design**: The code structure must use classes and functions to separate different functionalities like data loading, data processing, and performance evaluation.

Deliverables:

1. **Modular Python Code**:
   - Code implementing both sequential and parallel methods using pandas and Dask.
   - The code should follow OOP principles with clear, reusable components.
2. **Documentation**:
   - Instructions on how to run the code, explanations of the methods, and any dependencies required for the project.
   - Clear explanations of the performance results and how the time savings were achieved.
3. **Performance Comparison Report**:
   - A report comparing the execution times of the sequential and parallel processing methods.
   - Graphs or tables showing the time reductions achieved by using Dask for parallel processing.

Conclusion:

This project demonstrates the advantages of using Parallel and Distributed Processing techniques to significantly reduce the processing time for large datasets, such as the NYC Taxi Trip Dataset. By comparing sequential (pandas) and parallel (Dask) methods, the project illustrates how PDP techniques can scale up data analysis tasks, making them suitable for big data applications. The use of OOP design ensures that the solution is maintainable, reusable, and scalable, making it adaptable for similar big data challenges in various industries.

By the end of the project, you will have a clear understanding of the performance benefits of parallel computing and how it can be implemented in Python for real-world applications.