BIG DATA PROJECT

Report project

Fournier Pierre-Antoine 1820253067
Bichurina Sofia 1820253048
Dyoilis Anastasios 1820253075
Volkova Diana 1820253051

Contents

1 Introduction

In recent decades, the rapid growth and widespread adoption of digital technologies have fundamentally transformed the way information is generated, processed, and analyzed. This exponential increase in data volume, variety, and velocity—commonly termed "Big Data"—has introduced significant challenges and opportunities in data analytics. The sheer scale of modern datasets has rendered traditional data-processing techniques inadequate, prompting the emergence of powerful new frameworks designed explicitly for large-scale data analytics and visualization. At the forefront of these developments is Apache Spark, an open-source distributed computing framework renowned for its efficiency, scalability, and robust performance.

This project seeks precisely to bridge this usability gap. By combining the analytical strengths of PySpark with the user-friendly nature of Linux command-line tools, we provide users with straightforward, intuitive commands that internally perform powerful Big Data processing tasks. Specifically, our project implements commands that mimic traditional Linux utilities, facilitating the generation of insightful visual outputs such as pie charts, histograms, scatter plots, bar charts, and interactive geographical maps directly from simple user commands.

Our motivation stems from the recognition that effective data visualization is essential for clear communication and interpretation of complex information. Graphical outputs enable users to rapidly interpret patterns, trends, and correlations that may be difficult or impossible to discern from raw data alone. Interactive geographic visualizations further enrich user experience by providing context and immediate insights into spatial distributions. This approach empowers users of varying technical backgrounds to leverage Big Data analytics effectively, democratizing powerful analytical capabilities previously reserved for specialized data engineers.

Furthermore, this project contributes practical value through the exploration and analysis of publicly available taxi trip data. The ability to quickly visualize and interpret patterns such as payment method preferences, trip duration distributions, and geographic trip densities has meaningful implications for transportation planning, business analytics, and public policy decisions.

In this report, we detail our comprehensive approach, from architectural design and tool selection to practical implementation, visualization, and analytical results. We highlight the challenges encountered and discuss effective solutions, ultimately demonstrating the viability and practicality of integrating advanced Big Data analytics into an intuitive, familiar Linux command-line environment.

2 Group members and task assignements

The group members contributed as follows:

- Fournier Pierre-Antoine: PySpark implementation, command-line integration, script development
- Bichurina Sofia: Data parsing, querying, and batch processing
- Dyoilis Anastasios: Visualization development (NumPy, Matplotlib, Folium, OpenRouteService)
- Volkova Diana: Shell scripting, testing, and documentation

All members participated collectively in troubleshooting and refining the overall project.

3 System Design

3.1 Architecture and Functionalities

Our system consists of a modular architecture that includes:

- User Interface (CLI): Bash script providing simple Linux-style commands (e.g., `taxi pay 2009`).
- Data Parsing Module: Handles user inputs, selects appropriate .parquet files based on the year provided.
- Data Processing Module (PySpark): Executes distributed data queries and transformations, specifically selecting and grouping data.
- Visualization Module: Generates visualizations such as pie charts, bar charts, histograms, scatter plots, and heat maps.
- Mapping Module: Interacts with a local OpenRouteService server to draw interactive geographic maps using Folium.

This modular design allows independent development, easy troubleshooting, and scalability.

4 Platform & Tools

The project utilizes the following technologies and platforms:

- PySpark (Apache Spark): Distributed data processing
- Python: Main programming language for scripting and data analysis
- NumPy Matplotlib: Visualization of statistical data (pie charts, histograms, scatter plots, bar charts)
- Folium: Interactive geographic mapping and heatmaps
- OpenRouteService: Local server for geographical routing and polyline mapping
- Linux Shell (Bash): Command-line scripting and integration
- GitHub: Version control and collaboration

The combination of these tools provided an optimal balance of power, usability, and flexibility for data analysis and visualization tasks.

5 Implementation

The core of our project implementation involved developing commands similar to traditional Linux utilities that internally leverage PySpark to efficiently handle taxi trip data stored in local .parquet files. Users interact with the system through intuitive command-line commands such as:

```
taxi pay 2009
```

This command initiates a Bash script, which internally runs:

```
python3 -m main pay 2009
```

The script parses user input, dynamically selecting the appropriate .parquet dataset corresponding to the requested year. PySpark queries data using operations such as select and groupBy, transforming results into Python lists for visualization.

Visualization tasks employed NumPy and Matplotlib for creating statistical graphs, and Folium combined with OpenRouteService to produce interactive geographic maps. Batch processing was implemented to address memory management challenges, efficiently handling data in manageable segments.

6 Result & Demonstration

The implemented system successfully generated clear, intuitive visualizations:

- Pie Charts: Clearly illustrated payment method distributions for selected years.
- Histograms and Bar Charts: Showed frequency distributions and categorical analyses such as trip lengths and fare amounts.
- Scatter Plots: Revealed relationships between trip variables, enhancing insights into patterns within data.
- Heat Maps and Route Maps (Folium OpenRouteService): Offered interactive visualization of geographical data, allowing users to easily interpret taxi trip distributions and routes.

The visualizations effectively demonstrated the capabilities of PySpark integration and intuitive command-line interaction achieving our project's primary objectives.

7 Difficulties & Solutions

During the development and implementation of our project, several technical challenges arose, requiring innovative and practical solutions. This section details these difficulties and describes the strategies adopted to overcome them.

7.1 Memory Management Issues

One of the most significant challenges was managing memory efficiently while handling moderately large datasets with PySpark and visualization libraries. Initially, attempting to process large datasets entirely in memory resulted in frequent memory exhaustion errors. **Solution:** We addressed this issue through the implementation of batch processing techniques. By dividing the data into smaller, manageable segments or batches, we significantly reduced the memory footprint required during processing. Each batch was processed independently and sequentially, ensuring stability, scalability, and efficient resource utlization throughout data transformations and visualizations.

7.2 Complexity in Data Parsing and Selection

Selecting and correctly parsing data from '.parquet' files based on dynamic user inputs (such as different years) proved complex and error-prone. Misalignment or incorrect file selection initially resulted in inaccuracies and inconsistent outputs. **Solution:** A dedicated Python parsing functionwas implemented to dynamically select and filter datasets according to user-provided arguments. This function clearly and consistently handles input validation, selects the appropriate datasets, and integrates seamlessly with the rest of the PySpark workflow, greatly enhancing robustness and reliability.

7.3 Local OpenRouteService Integration

Integrating the local OpenRouteService server for interactive map generation posed logistical and integration challenges, particularly involving server setup, request handling, and error management. **Solution:** We successfully established a local instance of OpenRouteService separately, which allowed controlled and reliable API access. Python scripts were then designed to interact via HTTP requests with thelocal server instance, retrieving accurate routing information. Comprehensive error handling ensured robustness, improving the reliability of geographic visualizations such as polylines on interactive maps.

7.4 Visualization and Plotting Complexity

Generating clear, informative, and responsive visualizations required careful selection and integration of visualization libraries (NumPy, Matplotlib, Folium). Achieving consistency and clarity across multiple visualization types initially presented challenges. **Solution:** By leveraging NumPy for data preprocessing and Matplotlib for statistical visualizations, alongside Folium for geographic visualizations, we developed a clear, modular approach for visualization. Standardized plotting templates and careful tuning of parameters ensured consistency and clarity in graphical outputs, significantly enhancing interpretability and user experience. Overall, the solutions adopted for these challenges not only resolved immediate technical difficulties but also contributed to a robust, scalable, and highly maintainable system, demonstrating effective practices for Big Data project implementation.

8 Conclusion

The project successfully integrated powerful Big Data processing with a simple, user-friendly Linux command-line interface, achieving the goal of simplifying complex data analysis tasks. PySpark provided the necessary power, scalability, and efficiency to handle medium-sized datasets comfortably, while visualization libraries like NumPy, Matplotlib, and Folium provided intuitive, insightful visual outputs.

Our chosen approach demonstrated the effectiveness of combining PySpark with simple Linux commands, significantly enhancing usability and accessibility. Challenges such as memory management and data complexity were effectively solved through batch processing and modular design, improving overall robustness and scalability.

Overall, the project demonstrates a practical model for integrating Big Data analytics into familiar command-line environments, empowering users with powerful, easily accessible analytical tools for data-driven decision-making and exploration.