LOVELY PROFESSIONAL UNIVERSITY

RESULT MANAGEMENT SYSTEM USING HADOOP, SPARK, AND HIVE

Big Data Processing for Student Analytics

Submitted by
Amal Rosh Vinoy (12205422)

Department of Computer Science

1. Introduction

The Result Processing System leverages Hadoop, Spark, and Hive to manage and analyze large-scale student data efficiently. Traditional result management systems suffer from scalability and performance issues. This system mitigates these problems by utilizing Hadoop's HDFS for distributed storage, Spark for in-memory computations, and Hive for SQL-based queries. The workflow involves data ingestion into HDFS, MapReduce-based processing, Spark DataFrames transformation, and structured storage in Hive for analysis. The system also integrates visualization libraries like Matplotlib and Seaborn to generate insightful visual representations of student performance. Future improvements include real-time streaming with Kafka and an interactive dashboard for better user accessibility.

2. Technologies Used

- **Hadoop 2.7.7**: Distributed storage and processing for large datasets.
- **Apache Spark**: In-memory data processing for faster computations.
- **Hive**: SQL-based querying of structured student results.
- **HDFS (Hadoop Distributed File System)**: Efficiently manages student records.
- **Python (PySpark, Pandas, Matplotlib, Seaborn)**: Used for scripting, processing, and visualization.

3. Project Workflow

1. **Environment Setup**: Configured OpenJDK 8, Hadoop, and PySpark. Established key Hadoop files (hdfs-site.xml, core-site.xml) and set up NameNode and DataNode.
2. **Hadoop HDFS Configuration**: Created directories for raw student data (/input), processed results (/output), and user-specific files (/user/root). Uploaded student data for batch processing.
3. **MapReduce Processing**: Implemented Mapper (reads and processes student records) and Reducer (computes subject-wise averages). Executed Hadoop Streaming jobs to process and aggregate data.
4. **Data Processing in Spark**: Loaded student data into Spark DataFrames, applied transformations (filtering, aggregation), and store results in HDFS.
5. **Statistical Analysis and Visualization**: Computed essential metrics (mean, max, min) and generated visual reports (bar charts, histograms, box plots) using Matplotlib and Seaborn.
6. **Hive Integration and Query Execution**: Created Hive tables, and executed queries to identify top performers, pass/fail trends, and subject-wise analysis.

4. Explanation of Code Components

- **Mapper (mapper.py)**: Reads student records and emits key-value pairs (subject, marks).

- **Reducer (reducer.py)**: Computes subject-wise averages and stores results in HDFS.
- **Spark DataFrame Transformations**: Includes filtering null values, aggregation, and data partitioning.
- **Hive Queries**: Executes SQL-like queries to analyze student performance and trends.
- **Visualization Scripts**: Uses Matplotlib and Seaborn to create graphical reports.

5. Key Findings and Observations

- Efficiently processed large student datasets using Hadoop-Spark workflows.
- Improved performance through Spark's DataFrame API.
- Enabled structured insights through Hive queries.
- Visualized student performance trends using statistical graphs.

6. Challenges and Solutions

- **Encoding Issues**: Resolved UTF-8 errors by setting fallback encoding in the Mapper.
- **Data Format Errors**: Implemented exception handling in the Reducer to skip malformed inputs.
- **Performance Bottlenecks**: Optimized Spark processing through partitioning and caching.

7. Conclusion

This project highlights the significant role of Big Data technologies in modernizing result management systems. Integrating Hadoop, Spark, and Hive, ensures a scalable, accurate, and high-performance approach to processing large-scale student data. The system effectively utilizes distributed computing, in-memory processing, and advanced statistical analysis to derive meaningful insights into student performance. Additionally, incorporating visual analytics enhances the ability to identify trends and patterns in academic results. Looking ahead, future enhancements such as real-time result streaming, predictive analytics, and an interactive web-based dashboard will further refine data accessibility and decision-making processes.

8. Future Enhancements

- Real-time result streaming using Apache Kafka.
- Integration with Apache Sqoop for seamless data transfer.
- Development of a Flask/Django-based interactive dashboard for real-time visualization.