HADOOP PROJECT REPORT

1. Introduction

Big Data processing has become an essential part of modern computing. This project demonstrates the use of Hadoop for processing student marks data using the MapReduce programming model. The aim is to efficiently compute and analyze student marks using parallel distributed computing.

2. Project Objective

The objective of this project is to utilize Hadoop Streaming with Python scripts (Mapper and Reducer) to process and analyze student marks data stored in a CSV file.

3. Technologies Used

- Hadoop 3.3.6
- Python 3
- HDFS (Hadoop Distributed File System)
- MapReduce
- Kali Linux (for execution environment)
- Flask (for dashboard visualization)

4. Hadoop Components and Workflow

Hadoop consists of the following components:

- **HDFS (Hadoop Distributed File System):** Used for data storage.
- **MapReduce:** Used for parallel data processing.
- **YARN (Yet Another Resource Negotiator):** Manages resources and job scheduling.

Workflow:

1. Data is stored in HDFS.
2. The Mapper processes the data and generates intermediate key-value pairs.
3. The Reducer aggregates the data and provides the final output.
4. The output is stored in HDFS for further analysis.
5. The processed results are retrieved and displayed on a Flask-based dashboard.

5. Implementation Details

5.1 Data Source

The dataset used in this project is student_mark.csv, which contains student IDs, subject names, and marks.

Code

```
import random

import pandas as pd

from faker import Faker

import numpy as np

fake = Faker()

student = []


for i in range (1,10001):

    student_id = f"S{i:05d}"

    Student_name = fake.name()

    subjects = ["Electronics","Programming","Database","Data_Science","Math","English"]

    marks = np.random.randint(40,100,size = 6)

    student.append([student_id,Student_name]+list(marks))


df = pd.DataFrame(student, columns=["StudentId","Name","Electronics","Programming","Database","Data_Science","Math","English"])


df.to_csv('students_marks.csv',index = False)

print("Generated")
```

5.2 Mapper and Reducer Code

Mapper (``): Extracts relevant data and emits key-value pairs.

```
import sys
```

```
for line in sys.stdin:
    line = line.strip()  # Remove leading/trailing whitespace
    parts = line.split(',')  # Split by comma

    if parts[0].strip() == "StudentID":  # Handle header row
        continue

    student_id = parts[0].strip()
    name = parts[1].strip()

    try:
        marks = list(map(int, parts[2:]))  # Convert marks to integers
    except ValueError:
        continue  # Skip lines with invalid data

    total_marks = sum(marks)
    percentage = total_marks / len(marks)

    # Assign grade based on percentage
    if percentage >= 90:
        grade = "A"
    elif percentage >= 80:
        grade = "B"
    elif percentage >= 70:
        grade = "C"
    elif percentage >= 60:
        grade = "D"
    elif percentage >= 50:
        grade = "E"
    else:
        grade = "F"

    # Print results in tab-separated format
    print(f"{student_id}\t{name}\t{total_marks}\t{percentage:.2f}\t{grade}")
    if len(parts) == 3:
        student_id, subject, marks = parts
        print(f"{student_id}\t{marks}")
```

5.3 Hadoop Streaming Execution

Command to execute the job:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
  -input /user/data1/student_mark.csv \
  -output /user/output \
  -mapper /home/deep/python/haddop_s/mapper.py \
  -reducer /home/deep/python/haddop_s/reducer.py
```

Error Resolution

If the output directory already exists, remove it before execution:

```
hadoop fs -rm -r /user/output
```

6. Challenges and Solutions

- **File Not Found Error:** Verified that the input file was correctly placed in HDFS.
- **JAR File Issue:** Used the correct path /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar.
- **Output Directory Exists:** Removed the existing directory before running the job.

7. Results and Analysis

The system successfully computed the total marks of students from the CSV dataset and stored the results in HDFS. The output format is:

```
Student_ID   Total_Marks
101          250
102          280
...
```

8. Dashboard Visualization

To visualize the processed data, a Flask-based web dashboard is implemented. The dashboard retrieves output from HDFS and displays it in a structured format using Python's Flask framework.

9. Conclusion

This project demonstrated the use of Hadoop Streaming with Python for data processing. The MapReduce paradigm enabled efficient parallel processing, and the Flask-based dashboard provided a user-friendly visualization of the processed data.

10. Future Scope

- Expanding the system to handle larger datasets.
- Implementing real-time processing with Apache Spark.
- Enhancing the dashboard with interactive visualizations.

11. References

- Hadoop Official Documentation
- Python MapReduce Tutorials
- Flask Web Development Guides
- Big Data Processing Guides