(Group B2).

Experiment - 12

- Title of the assignment - Mapreduce.
- Problem statement - Design a distributed application using Mapreduce which processes a log file of a system.
- Pre-requisite - • Basics of java
  • Basics of Hadoop.
- Objective : Students must be able to use mapreduce effectively.

- THEORY :

1] Introduction.

- In modern systems, log files are extensively generated for monitoring, debugging and analysis.
- These files can become massive especially in distributed environments.
- Hadoop mapreduce provides a scalable and fault-tolerant method to process such large log files in parallel across multiple nodes or even on a local setup.

- This application will –

- Parse each line of a log file.
- Extract and count occurences of specific types of logs (eg. INFO, ERROR, WARN).
- Output the number of log entries by type.

2] Sample log file format.

```
2025-03-01  10:15 :23  INFO  System started successfully.
2025-03-01  10:16 : 00 WARN  low memory warining.
2025-03-01  10:17:05   ERROR Disk not found.
2025-03-01  10:18:15   INFO. Backup completed.
```

3] Map reduce workflow –

1. Mapper Logic
   - Reads each line
   - Extracts log level (eg. INFO, ERROR)
   - Emits key-value pair as (Loglevel 1).

2. Reducer logic.
   - Aggregates counts for each log level.
   - Outputs total occurences of each log type.

4] (a) Mapper class -

The mapper class is a crucial component in a mapreduce job responsible for processing each input record and generating intermediate key - value pairs.

In the context of processing log file, the mapper class parses each log entry and extracts relevant information. The typical steps involved in implementing a mapper class for log file processing include.

1. Input parsing - Read each line of log file.

2. Data extraction - Extract relevant information from each log entry, such as timestamps, error codes or other data points of interest.

3. Data transformation - Convert extracted information into key value pairs. For ex - if the goal is to analyze error frequencies, the mapper might emit <error-code 1> pairs for each occurence of an error code in log entry.

4. Output Emission - Emit the key-value pairs to the Map reduce framework for further processing.

The mapper class extends Mapreduce framework and overrides the map() method to define custom logic for processing input records.

(b). Reducer Class -

- It is a crucial component in a mapreduce job, responsible for aggregating and processing the intermediate key - value pairs. generated by mapper class.
- In the context of processing a log file, the reducer class receives key value pairs where the key represents a unique identifier (eg. an error) and value represents the count of occurences.

- The typical steps involved in implementing a reducer class for log file processing.

1. Input Aggregation - Receive key-value pairs grouped by key from map reduce framework.

2. Data aggregation - Aggregate the counts of occurences for each unique key.

3. Output generation - Produce the final output, which may include aggregated statistics, summaries, or any other desired analysis results.

- Reducer class overrides reduce() method.

* **Conclusion:** Thus we successfully implemented the distributed mapreduce for log file processing.