Experiment - 11

(Group B 1)

- Title of the Assignment – Hadoop for big data.

- Problem Statement – Write a code in java for a simple Word Count application that counts the number of occurrences of each word in a given input set using Hadoop mapreduce framework on local - standalone set up.

- Pre-requisites – Java language basics.

- Objective: Students must be able to use big data related. framework. and apply it.

- THEORY :

1) HADOOP.

- Hadoop is an open source framework developed by the apache foundation that allows for distributed storage and processing of large datasets using map reduce programming model.

- It is designed to run on commodity hardware and supports fault tolerance and scalability.

- The wordcount application is one of the simplest and most classic examples to demonstrate the map reduce paradigm.

- It processes input text files and counts the frequency of each word.

2] Key concepts and definitions -

Hadoop components

- HDFS (Hadoop distributed file system) - Stores large files across multiple machines.

- Mapreduce - A programming model for processing large datasets with a distributed algorithm on a cluster.

- Job tracker / resource manager - Co-ordinates jobs and resource allocation.

- Task tracker / Node manager - Executes tasks on individual nodes.

3] Map reduce components -

- Mapper - Processes input data and outputs key-value pairs.
- Reducer - Aggregates values based on keys from the mapper.
- Combiner - An optional component that acts like a mini reducer to optimize performance by reducing data transfer.

4] Working of word count program –

A. Input – A text file containing sentences for eg – hello-world.

B. Mapper phase –
- Input – A line of text.
- Process –
  - Split the line into words.
  - Emit each word with value 1.
- Output – Key value pairs – ("hello", 1)

C. Shuffle and sort phase –
- Groups all values by keys.
- Example – ("hello", [1, 1]), ("world", [1])

D. Reducer phase.
- Input – Key and list of values.
- Process –
  - Sum of all values for a key.
- Output – Final word count like ("hello", 2).

5] Local Standalone Mode –

- This mode is used for testing and development.
- The job runs on a single JVM without using HDFS.
- Useful for small datasets and debugging.

6] Applications of Word Count –

- Text mining and NLP.
- Analyzing logs and documents.
- Social media data processing.

7) Advantages of MapReduce –

- Scalable - Easily handles data in petabytes.
- Fault tolerant - Automatically manages hardware failures.
- Cost -effective - Runs on commodity hardware.

8] Step by step guide to run Wordcount in hadoop (Standalone mode) -

Pre-requisites - 1. Java JDK installed.
2. Hadoop installed and configured in local mode.
3. Environment variables - JAVA-HOME, HADOOP_HOME, PATH includes $ HADOOP-HOME /bin.

Step 1: Prepare your directory and files:
```
mkdir wordcount app.
cd wordcount app.
```

- Create sample input file: mkdir input.
- Save java code inside this file.

Step 2: Compile the wordcount java program:
```
export HADOOP- CLASSPATH = $ (hadoop classpath)

javac -classpath $HADOOP_CLASSPATH -d. Wordcount.java.
```

Create a JAR file - jar -cvf wordcount.jar *.class.

Step 3: Run the mapreduce Job.

```
hadoop jar wordcount.jar wordcount input output.
```

- Note: if output/ already exist, either delete it or change the output folder name.

- To delete previous output - `rm -r output.`

Step 4: View the output.

```
cat output/part-r-00000.
```

- You should see something like this -

```
hadoop      2
hello       2
mapreduce   1
world       1.
```

- **Optional** - Check files in output directory -
  `ls output/`

- Expected files:
  - part-r-00000 - The actual output.
  - _SUCCESS - Indicator that the job ran successfully.

* **Troubleshooting tips** -

- Ensure hadoop is in standalone/local mode in core-site.xml

- Run hadoop version to confirm Hadoop is installed properly.

- Ensure your input and output paths are accessible from your working directory.

- **Conclusion:** The wordcount example demonstrates the simplicity and power of Hadoop's Mapreduce programming model. It serves as a foundational exercise to understand distributed data processing and pares way for complex data intensive applications in Big data.

* * * * * * *