Experiment No.4
Experiment on Hadoop Map-Reduce
Date of Performance:
Date of Submission:

Aim: -To write a program to implement a word count program using MapReduce.

THEORY:

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. The implementation consists of three main parts:

1. Mapper

2. Reducer

3. Driver

Step-1. Write a Mapper

A Mapper overrides the ―map‖ function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides <key, value> pairs as the input. A Mapper implementation may output <key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number <line_number, line_of_text> . Map task outputs <word, one> for each word in the line of text.

Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

CONCLUSION:

Write pseudocode of Mapper() and Reducer()

Pseudocode for Mapper() Function

The Mapper() function processes input data and emits key-value pairs.

```
function Mapper(key, value):

    //Split the input value (e.g., a line of text) into words

    words = split(value, " ")
```

```
// For each word in the input
for each word in words:

    // Emit the word as the key and 1 as the value
    emit(word, 1)
```

Pseudocode for Reducer() Function

The Reducer() function aggregates the key-value pairs emitted by the Mapper() function.

```
function Reducer(key, values):
  // Initialize a counter for the sum of values
  total = 0

  // For each value in the list of values
  for each value in values:
      // Sum the values
      total = total + value

  // Emit the key and the total sum as the final output
  emit(key, total)
```