Ex No: 3 Implement word count program using Map Reduce.

AIM:

To implementing distinct word count problem using Map-Reduce

Algorithm :

The function of the mapper is as follows:

- Create a IntWritable variable 'one' with value as 1
- Convert the input line in Text type to a String
- Use a tokenizer to split the line into words
- Iterate through each word and a form key value pairs as Assign each work from the tokenizer (of String type) to a Text 'word'
- Form key value pairs for each word as < word,one > and push it to the output collector

The function of Sort and Group:

- After this, "aggregation" and "Shuffling and Sorting" done by framework. Then Reducers task these final pair to produce output.

The function of the reducer is as follows

- Initialize a variable 'sum' as 0
- Iterate through all the values with respect to a key and sum up all of them • Push

to the output collector the Key and the obtained sum as value For Example:

Example:

For the given sample input1 data file (input1.txt : Hello World Bye World) mapper emits: <Hello,1>

<World,1>

<Bye,1>

<World,1>

The second input2 data file (input2.txt : Hello Hadoop Goodbye Hadoop) mapper emits: <Hello,1>

<Hadoop,1>

<Goodbye,1>

<Hadoop,1>

WordCount also specifies a combiner. Hence, the output of each map is passed through the local combiner (which is same as the Reducer as per the job configuration) for local aggregation, after being sorted on the keys.

The output of the first map:

<Hello,1>

<Bye,1>

<World,2>

The output of the second map:

<Hello,1>

<Hadoop,2>

<Goodbye,1>

The Reducer implementation via the reduce method just sums up the values, which are the occurence counts for each key (i.e. words in this example).

Thus the output of the job is:

<Goodbye,1>

<Bye,1>

<Hello,2>

<Hadoop,2>

<World,2>

Python Implementation

mapper.py

```
#!/usr/bin/env python3
```

```
import sys

# Mapper Function
for line in sys.stdin:
    line = line.strip()             # Remove leading/trailing spaces
    words = line.split()            # Tokenize the line into words
    for word in words:
        print(f"{word}\t1")         # Emit key-value pair <word,1>
```

reducer.py

```
#!/usr/bin/env python3
import sys

current_word = None
current_count = 0
word = None

# Reducer Function
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

# Emit the last word
if current_word == word:
    print(f"{current_word}\t{current_count}")
```

To Run on Hadoop :

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-input input_data/ \
-output output_data/ \
-mapper mapper.py \
-reducer reducer.py
```

Expected Output:

```
Bye     1
Goodbye 1
Hadoop  2
Hello   2
World   2
```

Result:

Thus, the Word Count program using MapReduce was successfully implemented and executed to count the occurrences of each distinct word from multiple input files.