Assignment 5

Hadoop MapReduce:

Here I have implemented Map-Reduce application which uses two different files and counts the number of words available in both the files and creates the output directory consisting the text file having key-value pair with "word" as a key and "number of occurrence" of that word as a value. I have also used stop word file to eliminate the stop words. I have created a jar file and it is then run using following command.

Hadoop jar [path of .jar file] [classname] [/InputDirectory/file1.txt] [/InputDirectory/file2.txt] [/OutputDirectory] [-skip] [stopWord.txt]

Here "–skip" tag is used to determine the stop word file.

- So first of all I have tried to find the "-skip" tag so that I can get to know whether the stop word file exists or not in the argument.
- If it exists then it is stored in the cache so that we can use that file during mapping phase.
- Then in Mapper class(MyMapper class in my assignment) I have overridden the "setup()" method of the Mapper class, which sets the necessary things to perform the mapping function. It calls the "readSkipwordFile()" which reads the stop word file and stores it into the HashSet, so we can get the unique stop words without duplication of words.
- Now I have overridden the "map()" method of the Mapper class.
- "map()" method receives the data line by line from the specified text file(passed in the argument) available in the Hadoop input directory and then splits into the list of words so that we can check whether the word exists in the set of the stop words or not.
- If the word matches with any of the stop words then it is skipped and next word is checked, otherwise it is added to the list of map function output.
- This process is continued for all the words available in the text file.
- So now the mapping function is completed.
- Now I have created the "MyReducer" class which extends the in-built Reducer class.
- It simply reduces the number the words to its summation of occurrence.
- So this is how I have implemented MapReduce function.

❖ **Mapper's Input:**

<LongWritable,Text>

0, Hadoop is the Elephant King!

1, A yellow and elegant thing.

…..

❖ **Mapper's Output:**

<Text,IntWritable>

Hadoop, 1
Is, 1
…..

❖ **Reducer's Input:**

<Text,IntWritable>

Hadoop, [1]
Is, [1,1]
….

❖ **Reducer's Output:**

<Text,IntWritable>

Hadoop, 1
Is, 2
….