Hello World of MapReduce Framework - Java

Whenever any programming language needs to be explored, how to compile and run a program is described by so called “Hello World” program.

With respect to MapReduce Java programming with Hadoop Distributed File System support, “Word Count” is considered the simplest example to understand the paradigm and also how to compile, execute and verify results, etc.

Input files containing plain text need to be stored into HDFS location. This HDFS location will be provided to the program as a command line argument - input. Also, the second command line argument -input will be the location of HDFS where the resulting files - output have to be stored. Know that this output folder at the mentioned location will be created by the framework itself. If it exists already, the program will terminate before storing output.

Java program jar based map-reduce job will be submitted to ResourceManager everytime wordcount is expected for set of input files.

Framework reads line by line file text and provides a single line to mapper function as value in the argument for that function call.Understand that for every line mapper function is called. Actually mapper function is because of first class citizen characteristics and locality of reference optimization very efficiently in use once per line even. Ideally lines are expected to be longer enough and to need complex logic due characteristics of Big Data.

Mapper processes and generates set of key-value pairs containing word with frequency 1 for every words in the current line for WordCount program. Don't confuse with key-value and having unique key requirement. Every pair has its own independent existence.There can be multiple pairs having same key, in word count it is the word happening multiple times in the line possibly.

Mapper function of WordCount performs tokenization and generates plain simple pairs where every input word is associated as result of mapper <word,**1**> <key,**value**> showing just the fact that the current word has occurred 1 time just now irrespective of anywhere else in the line or files all over data.

Here, if for a different logic requirement this ‘1’ can be something else also. Instead of tokenization of value (line) any other logic can also be performed based on requirement.

After the execution of mapper, the framework behind the scene shuffles all resulting pairs and sorts to generate <key, **list of values**> kind of pair where all values for the same key is added

into values list. In word count program entries into values list will be number of 1's for a certain word, i.e. 'hello' has occurred 5 times overall then <'hello',[1,1,1,1,1]>.

For every such <key,list of values> reduce function is called which actually sums up all entries into the values list to produce total occurrences of that word. And finally the <word, count> pair will be dumped to the result file by multiple calls to mapper for various words.