Report on Map-Reduce program implementing Partitioner

- **Project Introduction:**
  Partitioner map reduce program calculates the number hits or request done by a particular ip address and the output had to be classified by months. In order to do so, it requires the implementation of partitioner and using only 12 required reducers for each of 12 months, thus breaking the standard MapReduce paradigm, which suggests all the values from a particular key will to the same reducer. On the other hand, in this example, ip addresses will go to multiple reducers.

- **Driver Description:**
  The driver class ProcessLogs comprises of the main method where job configuration has been defined. The main method in this driver class instantiates a job object which is configured by calling the setJarByClass method and specifying the jar file containing driver, mapper and reducer and reducer. The setJobName method is called to set the name for the job which is to appear in reports and logs. The setInputPaths and setOutputPaths is called on job object to set the input and output paths using command-line arguments. The setMapperClass and setReducerClass is called to specify the mapper and reducer classes. The input and output format of the key-value pair of is required to be specified well before running the Map Reduce program. In this case input format for the key of both mapper and reducer is same i.e. Text format. But the output format for value in mapper and reducer is different in partitioner project. Since mapper has output value in Text format and reducer has the output value in IntWritable.

  It is also required to set the number of reducer tasks to 12 using setNumReduceTasks method. Also partitioner class is set using setPartitionerClass method.

- **Mapper Description:**
  The mapper class includes a list of all the months to cross-check if indeed the months value is there in each of the line field in an expected format. The mapper class has a map method which when called takes the line field as a value parameter from the kv pair and splits in phrases or fields separated by spaces. In an expected format the month is expected to be at the third index of the string array generated by line field which is then written in HDFS as an mapper output using write method as a kv pair i.e., ip address as key, month as value.

- **Partitioner Description:**
  The partitioner is part of the program where all the output of the program is redirected to a specific reducer specified by a integer value which is already

stored in a hashmap with key as months and value as number representing the month and reducer the ip address will be directed to, for consolidation.

- **Reducer Description:**
  Upon the intermediate shuffling and sorting, all the all the values corresponding to the same integer value representing a reducer is consolidated and forwarded to reducer function. In this example, the reducer class i.e. CountReducer, has input (k,v) pair with key as ip addresses in Text Format and value as Iterable elements of type Text representing months which had to summed up and then averaged to get the counts or hits of all the ip addresses in a month. The output (k,v) pair contains key as letter in Text format and value as hits in IntWritable format, and is emitted by calling write method on context object.

- **DataFlow Description:**
  First off, the data to be processed is stored in Hadoop File System on the data node. The map task is used to process the data locally on each block where it is stored. The HDFS input file is retrieved in the form of k,v pair with key as byte offset where the line starts and values as line string itself. This (k,v) pair is mapped using the defined map function, which in this case is emitting mapper output key-value pair.

  Now further, before processing this intermediate key-value pair and using reduce function, come the role of the partitioner in this assignment. Partitioner is used to identify individual reducers each of the mapped key value pair need to go to for further processing. And then the reducer plays it's role by processing the intermediate consolidated data using the reduce function i.e. to sum up all the hits related to a particular ip address emitting the output key-value pair with key as the ip address and value as number of hits. This output (k,v) pair is stored for all 12 different reducers which was already defined in defined HDFS directory which can further be accessed, displayed and modified using command-line script.

- **Compile-Procedure Description:**
  The *.java files stored in the partitioner directory of the workspace is compiled using -javac command in the terminal. Once compiled all the *.class files are zipped in .jar file which further is run using hadoop jar command while also defining input directory for dataset and output directory for the processed data.

- **Test Data and Results:**
  The program is run and tested using Linux command-line scripts. It includes copying the data of local disk to hadoop file system using -cat while defining the directories where it is copying from and storing to. Further the driver, mapper and reducer classes are required to be compiled and exported as a .jar file. The next command runs the map reduce program assigning the tasks and processing the

data locally at the blocks. Further the output file partitioner/part-r-(0000 to 00012) of the MapReduce program can be accessed or displayed using script commands.

Report on Map-Reduce Program implementing a Custom WritableComparable

- **Project Introduction:**
  Writable program involves counting the number of times each pair string with lastname and firstname appears in a dataset. Now this project is required to implement the WritableComparable while defining readFields, write and comareTo methods.

- **Driver Description:**
  The driver class StringPairTestDriver comprises of the main method where job configuration has been defined. The main method in this driver class instantiates a job object which is configured by calling the setJarByClass method and specifying the jar file containing driver, mapper and reducer and reducer. The setJobName method is called to set the name for the job which is to appear in reports and logs. The setInputPaths and setOutputPaths is called on job object to set the input and output paths using command-line arguments. The setMapperClass and setReducerClass is called to specify the mapper and reducer classes. The input and output format of the key-value pair of is required to be specified well before running the Map Reduce program. In this case input format for the key of both mapper and reducer is same i.e. Text format. And the output format for value in mapper and reducer is in LongWritable format.
  The reducer class in this one is using LongSumReducer from Hadoop library class.

- **Mapper Description:**
  The mapper class StringPairMapper has a map function, which when called maps, the input k,v pair with key as offset where the line starts and value as the line itself, to an output k,v pair with key as StringPairWritable and value as 1. The process is done by splitting the input value in fields and retrieving them from string array.

- **StringPairWritable class Desccription:**
  StringPairWritable class is implemented with a constructor which is essentially empty and is required for serilization. There is another constructor which is

implement the value of the pair of string in the class variables left and right. The write menthod is used to serialize the fields left and right to out. Further readFields method is implemented to do the exact opposite i.e., deserialize the fields from in parameter.

The compareTo method is used to compare the left and the right string pairs with the left and right field of the other object.

- **DataFlow Description:**
  First off, the data to be processed is stored in Hadoop File System on the data node. The map task is used to process the data locally on each block where it is stored. The HDFS input file is retrieved in the form of k,v pair with key as byte offset where the line starts and values as line string itself. This (k,v) pair is mapped using the defined map function, which in this case is emitting mapper output key-value pair.

  Now further, before processing this intermediate key-value pair and using hadoop library defined reduce function LongSumReducer, k,v pair is sorted, shuffled and consolidated usingStringPairWritable class. The StringPairWritable class makes use of WritableComparable . And then the reducer plays it's role by processing the intermediate consolidated data using the reduce function i.e. to sum up all the values related to the same string name pair, emitting the output key-value pair with key as the string pair and value as number of repititions of them. This output (k,v) pair is stored in already defined in HDFS directory which can further be accessed, displayed and modified using command-line script.

- **Compile-Procedure Description:**
  The *.java files stored in the partitioner directory of the workspace is compiled using -javac command in the terminal. Once compiled all the *.class files are zipped in .jar file which further is run using hadoop jar command while also defining input directory for dataset and output directory for the processed data.

- **Test Data and Results:**
  The program is run and tested using Linux command-line scripts. It includes copying the data of local disk to hadoop file system using -cat while defining the directories where it is copying from and storing to. Further the driver, mapper and reducer classes are required to be compiled and exported as a .jar file. The next command runs the map reduce program assigning the tasks and processing the data locally at the blocks. Further the output file writable/part-r-0000 of the Writable MapReduce program can be accessed or displayed using script commands.