Project on Log File Analysis Map Reduce Program

a) **Project Introduction:**
Log File Analysis Map Reduce program calculates number of hit counts by each individual ip addresses which is logged in web log server file store in HDFS. This is done using a mapper, a reducer and a driver program.

b) **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. 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 namely, *LogFileMapper* and *SumReducer*.

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 type. And the output format of the mapper and reducer is of *IntWritable* type.

c) **Mapper Description:**
The mapper class in this example, i.e. *LogMapper*, is used to map the output key-value pairs resulting from input (k,v) pairs with key as byte-offset where the line starts, and value as the line string. When mapped, the output key is the substring of the line that represents ip address, with starting index 0 and ending index at the first index of ‘-’. The output value of the mapper is fairly simple, *i.e.* count 1.

d) **Reducer Description:**
Upon the intermediate shuffling and sorting, all the all the values corresponding to the same key is consolidated and forwarded to reducer

function. In this example, the reducer class i.e. *SumReducer*, has input (k,v) pair with key as ip address in Text Format and value as *Iterable* elements of type *IntWritable* which had to summed up. The output (k,v) pair contains key as letter in Text format type and value as sum of the counts in *IntWritable* format type, and is emitted by calling write method on context object.

e) **Data flow 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 key-value 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 where key is the ip address and value is the count 1.

Now further, before processing this intermediate key-value pair using reduce function, it is shuffled, sorted and consolidated. And then the reducer plays it’s role by processing the intermediate consolidated data using the reduce function to sum up all the counts related to a particular ip address emitting the output key-value pair with key as the ip address and value as number of counts of hits related to that ip address. This output (k,v) pair is stored in defined HDFS directory which can further be accessed, displayed and modified using command-line script.

f) **Program test procedure 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 *fileanalysis/part-r-00000* of the Map Reduce program can be accessed or displayed using script commands.