Project Report on Average Word Length

a) **Project Introduction:**
Average Word Length Map Reduce program calculates average word length of all the words, in a given excerpt of writing, that has the first letter as ‘a’, ‘b’, ‘c’ and so on. While calculating that case sensitive letters has been considered. This is done using a mapper, a reducer and a driver program.

b) **Driver Description:**
The driver class *AvgWordLength* 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.

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 average word length project. Since mapper has output value in IntWritable format and reducer has the output value in DoubleWriter format because average of word length is supposed to be a double type number.

c) **Mapper Description:**
The mapper class in this example, *i.e. LetterMapper*, 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 starting letter of individual words being split of the input line string and the output value is 1 every time for each word.

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. AverageReducer*, has input (k,v) pair with key as letters in Text Format and value as Iterable elements of type IntWritable which had to summed up and then averaged to get the average word length of all the words starting with a particular letter. The output (k,v) pair contains key as letter in Text format and value as average word length in DoubleWritable format, 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 letter, the word begins with 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 *i.e.* to sum up all the counts related to a particular letter and calculate the average emitting the output key-value pair with key as the letter with which the word begins and value as average number of times the word with same beginning letter appears. 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 *wordlengths/part-r-00000* of the Map Reduce program can be accessed or displayed using script commands.