David Hatcher

CIS4930

Dr. Zhang

9/26/2020

Assignment 1 Project Report

**Average Word Length**

1. This project was to get the average length of each word starting with each letter for Shakespeare’s works. The overall input was the works of Shakespeare and the output was a key pair value of each letter or number, case sensitive, as the key and the average length of the words starting with that character as the value.
2. The driver was configured with the job name “Average Word Length”, the jar class was set to AvgWordLength.class, the number of Mappers and Reducers are defaulted, it does not use a combiner, and the Input/Output Formats were set by the arguments of the terminal call. The Mapper class was set to LetterMapper, while the Reducer class was set to AverageReducer. The Mapper Output Key-Value classes are set to Text.class and IntWritable.class respectively. While the Reducer Ouput Key-Value class are set to Text.class and DoubleWritable.class respectively.
3. The Mapper is LetterMapper which takes in LongWritable key, Text value, and the Context object. The value is the line of text, the key is the byte offset of the line, and the context is what the object that contains the output key-value pairs. The mapper outputs the a key-value pair of (Text,IntWritable). To generate the key value pairs the mapper first takes the line that it is passed, splits it by nonword character (e.g. whitespace, punctuation) then adds the key-value pair of the first letter and the length of the word.
4. The Reducer is AverageReducer, which takes in a Text key, Iterable<IntWritable> values, and Context object. The value is the current letter, while the values are a list of the lengths of each word starting with that letter, and the context is the output values object. It outputs a Text object and a DoubleWritable object. To generate it’s key-pair values it initializes a variable sum and count at 0, then sums all the values for that letter and divides the sum by the count, after this it writes the key and this divided sum to the context as it’s key-pair value.
5. First the data from the input files it passed line by line into the mapper. Then the mapper will split the line by the whitespace/punctuation, grab the first letter of each word and the length of that word. Once it has these two values it writes to the context as a key-value pair, where the key is the first letter of the word and the value is the length of that word. Once this is all done it is put into the shuffle and sort process where all the values are split up by their keys so there would be k-v pairs in the form of (n,(2,3,4,5)) then these values are passed to the Reducer which will then take the sum of all those numbers for each letter and divide it by the number of numbers giving the average, then writes to the context a k-v pair with the first letter of the word and the average something like (n,3.5). Which is then written to the output file specified by the command line arguments.
6. To compile the program you must run two terminal commands:
   1. javac -classpath `hadoop classpath` stubs/\*.java
   2. jar cvf AvgLength.jar stubs/\*.class
7. To run the MapReduce job and access the results you enter the following terminal commands:
   1. hadoop jar AvgLength.jar stubs.AverageWordLength shakespeare wordlengths
   2. hadoop fs -cat wordlengths/part-r-00000 | less

**Log File Analysis**

1. This project was to get the number of times each each IP address accessed a website. The overall input file was a log file containing the IP address, the date, the command given, the response from the server, and another value that I am not sure of but was not needed for this count.
2. The driver was configured with the job name “Process Logs”, the jar class was set to ProcessLogs.class, the input and output paths were specified by terminal arguments, the Mapper class was set to LogFileMapper, the reducer class was set to Sum Reducer, and the mapper and reducer output key-value pairs were set to Text and IntWritable respectively. The number of mappers and reducers was defaulted.
3. The mapper takes an input of LongWritable key, Text value, and Context object. The key is the byte offset of the line the mapper is currently taking, and the value is the content of that line. It outputs a key-value pair with a Text object as the key and an IntWritable as the value. To produce the correct key-value pairs it splits the line by whitespace and grabs the first value of that array as the IP address, since that is the formatting for each line, IP address followed by a space then the other information. It will then write a k-v pair to the context of that IP address and the number 1.
4. The Reducer is a sum reducer. It receives a Text key, Iterable<IntWritable> values, and the Context Object. The key is the IP address, while the values is a list of the of 1s representing each time the IP address requested some content from the server. It then adds all the ones together and writes to the context a k-v pair with the IP address as the key and the total number of hits as the value.
5. First the data is read line by line from the input file and passed into the mapper by each line. Then the mapper will split the line by whitespace, grab the IP address out of it and output a k-v pair in the form (IP address, 1). Once this is done it will go to the shuffle and sort process where all the IP addresses are grouped together and send to the reducer as k-v pairs (e.g. (192.186.1.1,(1,1,1,1,1)). The reducer then takes these k-v pairs and sums them together to create a k-v pair containing the IP address and the total number of hits (e.g. (192.168.1.1,5)). Which is then written to the output file specified by the command line arguments.
6. To compile this program you must run two terminal commands:
   1. javac -classpath `hadoop classpath` stubs/\*.java
   2. jar cvf logfile.jar stubs/\*.class
7. To run the MapReduce job and access the results you enter the following command in the terminal:
   1. hadoop jar logfile.jar stubs.ProcessLogs weblog iplogging
   2. hadoop fs -cat iplogging/part-r-00000 | less