MapReduce 2: COMP41111 Cloud Computing

Linda Smith 17202469

November 23, 2018

Abstract

A common task in digital forensics is the analysis of log files. Each line (entry) of a web server's log file normally contains important information such as an IP address, date and time of request, request line, HTTP status, etc. The objective of this assignment was to create a Map/Reduce programming model in Python or Java to analyse a file's components to extract information from the log files.

1 Question 1

Define the input and the output of the Map function(s) for each programming problem. Justify your answer.

1.1 Approach

My approach to this problem was to create 1 mapper function and 3 reducer functions in python. The mapper takes the file (access_log) as command line input, extracts the IP addresses using a regular expression and returns an intermediate key-value pair of all the IPs mapped with the value '1'.

The mappers stdout is piped into linux's 'sort' command which shuffles/sorts the IPs into groups of their occurrence. The results of this sort is subsequently piped into each reducer function which execute an operation to find:

- Total no. of connections to the server i.e. server entries
- No. of distinct IPs
- No. of entries for each IP

1.2 Mapper function

def **Mapper**(path_to_file)

Input: access_file (server log file) from command-line

Output: Valid IPs mapped in key-value pairs e.g.

195.84.101.164 1

195.84.101.164 1

202.155.247.214 1

For the purpose of this exercise it was only necessary to make one mapper as the reducers are all concerned with the information gained from mapping IP addresses from the log file. Additional mappers would be necessary if we were concerned with what browser was associated with each IP instance or we wished to extract the type of HTTP status code generated, or the type of request method.

1.3 Reducer functions

def **Reducer1()**
""" Returns the total number of of connections to the server i.e. total server entries """

Input: Receives stdout from mapper function

Output: No. of total server entries: 4224

def **Reducer2()**

""" Returns the total number of of connections to the server i.e. total server entries """

Input: Receives stdout from mapper function

Output: returns the number of distinct IPs: 117

def **Reducer3()**

""" Returns the total number of of entries for each IP in the server log """

Input: Receives stdout from mapper function

Output: total number of of entries for each IP in the server log

The reducer3's output would look like:

213.154.246.195 15

213.222.28.155 28

213.84.219.101 881

216.39.192.242 8

217.169.51.254 3

221.201.1.230 4

2 Python code implementation

```
Lindas-MBP:cloudComputing lindasmith$ ./mapper.py access_log | sort | ./reducer1.py
4224
Lindas-MBP:cloudComputing lindasmith$ ./mapper.py access_log | sort | ./reducer2.py
117
Lindas-MBP:cloudComputing lindasmith$ ./mapper.py access_log | sort | ./reducer3.py
128.30.52.13 3
128.30.52.34 12
131.211.84.18 2
133.27.228.132 12
145.99.163.83 6
150.5.65.193 6
193.95.95.254 13
195.159.7.242 4
195.84.101.164 7
200.198.94.211 7
201.7.100.194 6
202.155.247.214 1
202.155.247.215 1
207.22.48.162 9
212.126.165.242 16
212.177.56.179 13
212.238.250.74 7
213.154.246.195 15
213.222.28.155 28
213.84.219.101 881
216.39.192.242 8
217.169.51.254 3
221.201.1.230 4
57.66.55.2 6
58.177.114.146 1
62.2.213.34 5
62.234.138.228 105
64.233.172.18 2
64.233.172.21 3
64.233.172.4 1
66.249.65.1 12
66.249.65.109 89
```

Figure 1: Cropped output from running the map/reduce code.

3 Future improvements

The speed of the task execution could be greatly improved by parallelization. The 3 operations performed by the mapper and the reducers could be coordinated by running them in parallel.

If I was to repeat the assignment I would write the code in Java. As the concept of map reduce was orignally conceived and designed in Java they have created interfaces for mappers and reducers which can be implemented using classes from Hadoop's package. For a single node set-up, Java will automatically handle parallelisation depending on the number of processing cores you have.