Pipelined Data Processing Using Spark

Chan-Ching Hsu

Apache Spark, a fast and general engine for big data processing, provides a basic data abstraction called RDD (Resilient Distributed Dataset), which is a collection of elements, stored in a distributed manner across a cluster. Using RDD transformations, Spark is well suited for data processing in pipelines. Spark also provides rich APIs for RDDs so that users can easily operate data in parallel. In addition, users can optionally persist RDDs in memory so that it will speed up computing when reusing the data.

WordCount Example

The "WordCount" java code first uses **flatMap()** method to split each line of text into words and each word is as one element in the RDD. Then it uses **mapToPair()** method to transform RDD into PairRDD, that converts each element into <key, value> pair where key is the word and value is one. Finally, it used **reduceByKey()** method to sum all the ones to get the number of counts for each word. Note that the function in **reduceByKey()** method is applied in associative manner, like the Combiner in Hadoop.

For all the APIs on RDD and PairRDD, check the links to their javadocs:
https://spark.apache.org/docs/1.5.2/api/java/org/apache/spark/api/java/JavaRDDLike.html
https://spark.apache.org/docs/1.5.2/api/java/org/apache/spark/api/java/JavaPairRDD.html

For more examples in Java:
https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples

Task 1

Modify the word count example so that the output is sorted by the number of counts in descending order. The following is the snapshot of first 10 lines of the output based on the Gutenberg corpus as the testing input.

```
(9499589,the)
(5960882,of)
(5157312,and)
(4726874,to)
(3388519,a)
(2813328,in)
(1866070,that)
(1702990,I)
(1611990,was)
(1442544,he)
```

Task 2

Given two input files:

ip_trace – An Ip trace file having information about connections received from different source IP addresses, along with a connection ID and time. The format of IP trace file is:

<Time> <Connection ID> <Source IP> ">" <Destination IP> <Protocol> <Protocol-dependent Data>

raw_block – A file containing the connection IDs that were blocked. The format of block file is:

<Connection ID> <Action Taken>

The task is to regenerate the log file by combing information from other logs that are available. The lost firewall log should contain details of all blocked connections and should be in the following format.

<Time> <Connection ID> <Source IP> <Destination IP> "Blocked"

A. Regenerate the firewall file containing details of all blocked connections. The snapshot of first 10 lines of the generated firewall log is as follows.

```
1:43:20:149 6200148 123.73.107.72 135.50.129.20 Blocked
1:43:20:266 6200265 245.221.135.46 122.119.230.221 Blocked
1:43:20:332 6200331 40.111.90.113 76.16.116.183 Blocked
1:43:20:365 6200364 12.168.62.143 230.191.175.115 Blocked
1:43:20:554 6200553 54.32.69.151 185.81.43.192 Blocked
1:43:20:611 6200610 190.139.161.229 56.92.12.121 Blocked
1:43:20:686 6200685 18.125.206.93 13.49.123.136 Blocked
1:43:20:806 6200805 11.141.111.196 185.81.43.192 Blocked
1:43:20:824 6200823 185.170.96.137 67.221.87.66 Blocked
1:43:20:866 6200865 132.151.87.6 81.103.86.76 Blocked
```

B. Based on the previous program, generate a list of all unique source IP addresses that were blocked and the number of times that they were blocked. This list

should be sorted by the number of times that each IP was blocked in descending order. Below is the snapshot of first 10 lines of the output file.

```
(9798,108.2.235.200)
(9788,13.110.126.21)
(9776,63.101.203.191)
(9772,107.68.208.3)
(9769,79.60.169.28)
(9755,141.2.188.213)
(9751,2.164.178.163)
(9750,152.244.120.111)
(9746,216.91.116.228)
(9743,104.230.13.142)
```