Lab1 - MapReduce, HDFS, and HBase

Adam Hasselberg, adamhas@kth.se — Velisarios Miloulis, miloulis@kth.se

September 2018

1 Introduction

In this lab we construct a MapReduce program which reads a an XML file from HDFS and uses MapReduce to extract and compiles a TopTen list from the original data. In the Implementation section we go over how the program works. In Problems section we discuss two problems we encounter, with sample code and a trace of the result. In the Results section we show our final result.

2 Implementation

The implementation consists of 3 parts: Driver, Mapper and Reducer. The driver is pretty straight forward, it sets up the parameters for the Job and then starts it, and creates the Input Splits.

The Mapper is more complex. It reads the XML file line by line, which could be distributed to several nodes, and for each line it splits the line up using quotation marks. A line from the XML looks like this:

```
<row Id="2442" Reputation="1"
 CreationDate="2014-07-10T19:37:13.557"
 DisplayName="Ace"
 LastAccessDate="2015-10-24T14:42:21.253"
 Views="1" UpVotes="0" DownVotes="0"
 ProfileImageUrl="https://www.gravatar.com/avatar/
 11d9881524b28ad93b91896cae605b27?s=128&amp;d=identicon&amp;r=PG"
 Age="40" AccountId="3123564" />
```

Each line is a separate record, and is split around quotation marks in such a way that each tag such as "Id" is a key and its value is mapped as the key's value using the given `transformXmlToMap` function, and each key-value pair of a single record is stored in a HashMap.

For each record, the Mapper keeps key-value pairs the user reputation and the whole record as Text in a TreeMap data structure. The specific data structure keeps its entries sorted on its keys, so in the `cleanup` function, the Mapper emits to the Reducer the top ten value entries of the TreeMap, i.e the top ten

records with the highest reputation. This makes the program run more efficiently when using many nodes as the final reduce phase will at most receive N*10 entries where N is the number of Mapper nodes.

The single Reducer is executed in a single node. It receives the top records from each Mapper and, similarly to the first step of the Mapper, it makes use of the given `transformXmlToMap` function to break down each record into a HashMap of tag and tag value key-value pairs. The Reducer also uses a TreeMap structure, but to hold the reputation and id values of each record. After all the records have been processed and their id and reputation values are stored into the TreeMap, proceeds to write each of the top ten reputation pairs in an HBase database.

3 Problems

We encounter two problems with the code. Both relate to the TreeMap the assignment constrains us to use. A major flaw of the program which we assume to be a latent bug of the given code in the assignment, is that the mapper uses reputation as key in a map. This means that if two users would share the same reputation only one of them would exist in the code. A lesser flaw, which we believe is a result of our runtime environment, is that the Text object used to store intermediate data does not function as expected when put into Map data-structures. When executed several consecutive Poll or Get calls to the map return the same Text object regardless of the content of the Map. Sample code and its resulting output:

```
for (org.apache.hadoop.io.Text t : values) {
  String s = t.toString();
  int k = Integer.parseInt(s.substring(0, s.indexOf(";")));
  LOG.info("****Intermediate step checking: " + t.toString()
  + " KEY = " + k);
  repToRecordMap.put(k, t);
}
for (int k : repToRecordMap.descendingKeySet()) {
LOG.info("****Finally adding: " + repToRecordMap.get(k).toString()
+ " KEY: " + k);
}
```

```
INFO topten.TopTen: ****Intermediate step checking: 1846;836 KEY = 1846
INFO topten.TopTen: ****Intermediate step checking: 1878;9420 KEY = 1878
INFO topten.TopTen: ****Intermediate step checking: 2127;108 KEY = 2127
INFO topten.TopTen: ****Intermediate step checking: 2131;434 KEY = 2131
INFO topten.TopTen: ****Intermediate step checking: 2179;84 KEY = 2179
INFO topten.TopTen: ****Intermediate step checking: 2289;548 KEY = 2289
INFO topten.TopTen: ****Intermediate step checking: 2586;21 KEY = 2586
INFO topten.TopTen: ****Intermediate step checking: 2824;11097 KEY = 2824
INFO topten.TopTen: ****Intermediate step checking: 3638;381 KEY = 3638
```

```
INFO topten.TopTen: ****Intermediate step checking: 4503;2452 KEY = 4503
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 4503
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 3638
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2824
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2586
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2289
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2179
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2131
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 2127
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 1878
INFO topten.TopTen: ****Finally adding: 4503;2452 KEY: 1846
```

The problems mentioned cost us quite some time, since there was no apparent bug in the code, but were overcome using different methods to parse the TreeMap.

4 Results

To run the program we follow the instructions given in the assignment instructions lab1.pdf page 13. We get the final result of the program by scanning our local hbase database

```
hbase(main):003:0> scan 'topten'
ROW                    COLUMN+CELL
 \x00\x00\x00\x00      column=info:id, timestamp=1537620111636, value=2452
 \x00\x00\x00\x00      column=info:rep, timestamp=1537620111636, value=4503
 \x00\x00\x00\x01      column=info:id, timestamp=1537620111636, value=381
 \x00\x00\x00\x01      column=info:rep, timestamp=1537620111636, value=3638
 \x00\x00\x00\x02      column=info:id, timestamp=1537620111636, value=11097
 \x00\x00\x00\x02      column=info:rep, timestamp=1537620111636, value=2824
 \x00\x00\x00\x03      column=info:id, timestamp=1537620111636, value=21
 \x00\x00\x00\x03      column=info:rep, timestamp=1537620111636, value=2586
 \x00\x00\x00\x04      column=info:id, timestamp=1537620111636, value=548
 \x00\x00\x00\x04      column=info:rep, timestamp=1537620111636, value=2289
 \x00\x00\x00\x05      column=info:id, timestamp=1537620111636, value=84
 \x00\x00\x00\x05      column=info:rep, timestamp=1537620111636, value=2179
 \x00\x00\x00\x06      column=info:id, timestamp=1537620111636, value=434
 \x00\x00\x00\x06      column=info:rep, timestamp=1537620111636, value=2131
 \x00\x00\x00\x07      column=info:id, timestamp=1537620111636, value=108
 \x00\x00\x00\x07      column=info:rep, timestamp=1537620111636, value=2127
 \x00\x00\x00\x08      column=info:id, timestamp=1537620111636, value=9420
 \x00\x00\x00\x08      column=info:rep, timestamp=1537620111636, value=1878
 \x00\x00\x00\x09      column=info:id, timestamp=1537620111636, value=836
 \x00\x00\x00\x09      column=info:rep, timestamp=1537620111636, value=1846
10 row(s) in 0.4030 seconds
```