Data Intensive Computing – Lab Assignment 1

Adriano Mundo, Riccardo Colella (Group 29)

1st October 2020

The purpose of this assignment is to show our proposed solution for the *MapReduce* program written using HDFS and HBase. We were given a list of user records in the *users.xml* file. We were asked to retrieve the top ten users with respect to their reputation. The following image shows a snippet from the xml data.
The goal is to retrieve the information of the top ten users based on their reputation

```
<row Id="-1" Reputation="1"
  CreationDate="2014-05-13T21:29:22.820" DisplayName="Community"
  LastAccessDate="2014-05-13T21:29:22.820"
  WebsiteUrl="http://meta.stackexchange.com/"
  Location="on the server farm" AboutMe="..;"
  Views="0" UpVotes="506"
  DownVotes="37" AccountId="-1" />
```

1. Getting started

In order to run the application, the following instructions are given.

1. Firstly, you need to have HDFS and HBase successfully installed.
2. You need to start HDFS NameNode and DataNode. Then create a folder input in HDFS, and upload files in it.

```
$HADOOP_HOME/bin/hdfs --daemon start namenode
$HADOOP_HOME/bin/hdfs --daemon start datanode

cd src/id2221/topten
$HADOOP_HOME/bin/hdfs dfs -mkdir -p topten_input
$HADOOP_HOME/bin/hdfs dfs -put data/users.xml
$HADOOP_HOME/bin/hdfs dfs -ls topten_input
```

3. You need to start the HBase and the HBase shell.

```
$HBASE_HOME/bin/start-hbase.sh
$HBASE_HOME/bin/hbase shell
```

4. Then you need to create the HBase table *topten* with one column family *info* to store the id and the reputation of users

```
create 'topten', 'info'
```

5. You also need to set some environment variables.

```
export HADOOP_CLASSPATH=$($HADOOP_HOME/bin/hadoop classpath)
export HBASE_CLASSPATH=$($HBASE_HOME/bin/hbase classpath)
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HBASE_CLASSPATH
```

6. You may want to change directory to the *src* folder and make a target directory, *topten_ classes* to keep the compiled files. Then, you can compile the code and make a final *jar* file.

```
cd src/id2221

mkdir topten_classes

javac -cp $HADOOP_CLASSPATH -d topten_classes topten/TopTen.java

jar -cvf topten.jar -C topten_classes/ .
```

7. At this point you can finally run the application.

```
$HADOOP_HOME/bin/hadoop jar topten.jar id2221.topten.TopTen topten_input topten_output
```

8. After the application run, you can check the results through the HBase shell with the following command.

2. Mapper Class

The main idea behind the *mapper class* is to read the data used as input from HDFS determining the top ten records of each of the input split, which is derived from the input file. The *map* process read each input records, generate the *<key, value>* pairs and then output the top records to the reduce phase, using the *cleanup* method.

The map function receives the input as String value, then use the helper function provided in the assignment *transformXmlToMap* to store a map of user reputation to the record. Then, it gets the id and the reputation of the user (processed record) as String.
After that, it checks and processes only the rows with id not equals to null. It uses the *TreeMap* structure to store the value retrieved from the intermediate map using the reputation (converting it to Integer) as key and the information as value, since the *TreeMap* is sorted on the key.
Finally, it checks if the size of the map is greater than ten hence it filters the top ten records and removes the record with the lowest reputation number using the method *firstkey*.

Finally, the *map* function passes its result to the *cleanup* method which is responsible to write the result to the *context*, used as input for the *Reducer Class*.

3. Reducer Class

The *reducer class* job is to take the input from the mappers and write the result in the HBase table. After that, it's possible to use the *scan* method in HBase shell to show the 10 rows selected.
The reducer class has a similar code to the mapper class. In the first part it uses an intermediate map and the helper function to store the data from the *Iterable* values. After that, it put the

record data in the *TreeMap* structure, using again the reputation (as Integer) as key of the map and checks if the size is greater than then.
The reducer class is configured to use just one reducer for handling data sharing between mappers with *job.setNumReduceTasks(1)*. This makes possible to iterate over the values.
Once stored the data in the *TreeMap* we can iterate again over the values storing the Reputation and Id as String.
The final part is to create the *HBase table* and add the data to the colums *info:rep* and *info:id* of the *topten* table.
Finally, it writes the data using the *context.write*.

```
scan 'topten'
```

4. Results

In the following figure the final results generated in the HBase shell are shown.
The highest Reputation value is 4503 for the user 2452, and the 10th is 1846 for the user 836.

```
hbase(main):006:0> scan 'topten'
ROW                   COLUMN+CELL
 1846                 column=info:id, timestamp=1601570603876, value=836
 1846                 column=info:rep, timestamp=1601570603876, value=1846
 1878                 column=info:id, timestamp=1601570603876, value=9420
 1878                 column=info:rep, timestamp=1601570603876, value=1878
 2127                 column=info:id, timestamp=1601570603876, value=108
 2127                 column=info:rep, timestamp=1601570603876, value=2127
 2131                 column=info:id, timestamp=1601570603876, value=434
 2131                 column=info:rep, timestamp=1601570603876, value=2131
 2179                 column=info:id, timestamp=1601570603876, value=84
 2179                 column=info:rep, timestamp=1601570603876, value=2179
 2289                 column=info:id, timestamp=1601570603876, value=548
 2289                 column=info:rep, timestamp=1601570603876, value=2289
 2586                 column=info:id, timestamp=1601570603876, value=21
 2586                 column=info:rep, timestamp=1601570603876, value=2586
 2824                 column=info:id, timestamp=1601570603876, value=11097
 2824                 column=info:rep, timestamp=1601570603876, value=2824
 3638                 column=info:id, timestamp=1601570603876, value=381
 3638                 column=info:rep, timestamp=1601570603876, value=3638
 4503                 column=info:id, timestamp=1601570603876, value=2452
 4503                 column=info:rep, timestamp=1601570603876, value=4503
10 row(s) in 0.2590 seconds

hbase(main):007:0>
```