2020-09-29

ID2221 – Data Intensive Computing

Lab 1 – MapReduce, HDFS, HBase, Spark

Nagasudeep Vemula<vemula@kth.se >

Akhil Yerrapragada<akhily@kth.se >

Overview

Here we would like to explain, design decisions taken when implementing **TopTen.java** and output generated when executing the file. The key functionalities provided are TopTenMapper, TopTenReducer and Driver. Now, we delve into each of their implementation in detail.

TopTenMapper:

First, we have used **transformXMLToMap** inside **map** function to get the information of the record and verify if the record has null **Id**. If the record does not contain null Id, we insert the Reputation value it contains to the TreeMap, **repToRecordMap.**

Second, Inside the **cleanup** function, since the **TreeMap** stores all the keys in ascending order, we have used **.descendingmap()** function to get records of users with highest reputation. Using **.entrySet()** function, we have converted the **TreeMap** to **Entry** to iterate over the content. We have used a helper variable and counter to get top ten records and **context.write** to emit those records to Reducer with **NullWritable** Key.

```
// Stores a map of user reputation to the record
TreeMap<Integer, Text> repToRecordMap = new TreeMap<Integer, Text>();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
    if (!transformXmlToMap(value.toString()).isEmpty() && transformXmlToMap(value.toString()).get("Id") != null ) {
        repToRecordMap.put(Integer.parseInt(transformXmlToMap(value.toString()).get("Reputation")), new Text(value));
    }
}

protected void cleanup(Context context) throws IOException, InterruptedException {
    // Output our ten records to the reducers with a null key
    int i = 0;
    for (Map.Entry<Integer, Text> entry : repToRecordMap.descendingMap().entrySet()) {
        if (i++ < N) {
            context.write(NullWritable.get(), entry.getValue() );
        }
    }
}
```

TopTenReducer:

The reducer receives the values (Top Ten user records) from Mapper, Iterates through each of them. In each iteration, we use **transformXMLToMap** to extract the user reputation and insert it to the provided **TreeMap.**

Similar to what we did in mapper, we use **.descendingmap()** and **.entrySet()** to get users with highest reputation and iterate through them. In each iteration, we add the key to "**rep**", value to "**id**" and write them to Hbase using **context.write** function.

```
private TreeMap<Integer, Text> repToRecordMap = new TreeMap<Integer, Text>();

public void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
    for (Text value : values) {
        repToRecordMap.put( Integer.parseInt(transformXmlToMap(value.toString()).get("Reputation")) , new Text( transformXmlToMap(value.toString()).get("Id") ) );
    }

    int i = 0;
    for (Map.Entry<Integer, Text> entry : repToRecordMap.descendingMap().entrySet()) {
        Put insHBase = new Put(Integer.toString(i++).getBytes());
        insHBase.addColumn(Bytes.toBytes("info"), Bytes.toBytes("rep"), Bytes.toBytes(entry.getKey().toString()));
        insHBase.addColumn(Bytes.toBytes("info"), Bytes.toBytes("id"), Bytes.toBytes(entry.getValue().toString()));
        context.write(NullWritable.get(), insHBase);
    }
}
```

The driver code below is responsible for all the required configuration

```
    Configuration conf = HBaseConfiguration.create();

    // define scan and define column families to scan
    Scan scan = new Scan();
    scan.addFamily(Bytes.toBytes("info"));
    Job job = Job.getInstance(conf, "top_rep");
    job.setJarByClass(TopTen.class);
    job.setMapperClass(TopTenMapper.class);
    job.setReducerClass(TopTenReducer.class);
    job.setMapOutputKeyClass(NullWritable.class);
    job.setMapOutputValueClass(Text.class);
    job.setNumReduceTasks(1);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    TableMapReduceUtil.initTableReducerJob("topten", TopTenReducer.class, job);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
```

Output:

The below output contains ten reputed users in descending order

```
hbase(main):001:0> scan 'topten'
ROW                                   COLUMN+CELL
 0                                    column=info:id, timestamp=1601328471679, value=2452
 0                                    column=info:rep, timestamp=1601328471679, value=4503
 1                                    column=info:id, timestamp=1601328471679, value=381
 1                                    column=info:rep, timestamp=1601328471679, value=3638
 2                                    column=info:id, timestamp=1601328471679, value=11097
 2                                    column=info:rep, timestamp=1601328471679, value=2824
 3                                    column=info:id, timestamp=1601328471679, value=21
 3                                    column=info:rep, timestamp=1601328471679, value=2586
 4                                    column=info:id, timestamp=1601328471679, value=548
 4                                    column=info:rep, timestamp=1601328471679, value=2289
 5                                    column=info:id, timestamp=1601328471679, value=84
 5                                    column=info:rep, timestamp=1601328471679, value=2179
 6                                    column=info:id, timestamp=1601328471679, value=434
 6                                    column=info:rep, timestamp=1601328471679, value=2131
 7                                    column=info:id, timestamp=1601328471679, value=108
 7                                    column=info:rep, timestamp=1601328471679, value=2127
 8                                    column=info:id, timestamp=1601328471679, value=9420
 8                                    column=info:rep, timestamp=1601328471679, value=1878
 9                                    column=info:id, timestamp=1601328471679, value=836
 9                                    column=info:rep, timestamp=1601328471679, value=1846
10 row(s) in 0.2690 seconds
```