Illustration of topten.java

1. Way of Design:

In part I, we are assigned to sort topten userId with the highest reputation in MapReduce way. In Map, we separate the input file, i.e., users.xml into serveral mappers and each mapper generate top N in their own list. In Reduce part, class TopTenReducer finds global topten userId.

In order to explain the process of MapReduce in a more reader friendly way, here we provide a metaphor. In order to select the topten swimmers in the world, we ask each country to select their own topten swimmers. This is what Map does. Then, we invite the selected swimmers from each country to attend an Olympic Games where we can select the topten swimmers all over the world. This is what Reduce does.

Figure 1 demonstrates how MapReduce work in topten.java

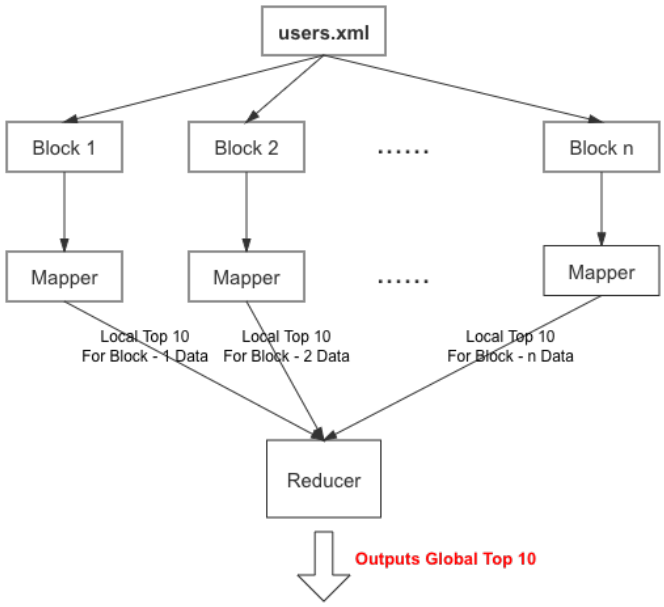

Figure 1 The Demonstration of MapReduce in topten.java

2. Functions in topten.java:

The function of public static Map<String, String> transformXmlToMap(String xml) split the input file into keys and values. For example, the first row will be splitted into the following table

<row Id="1" Reputation="101" CreationDate="2014-05-13T22:58:54.810" DisplayName="Adam" LastAccessDate="2016-06-10T22:48:51.523" Location="New York, NY" AboutMe="

<p>Full-time Internet. Gamer. Developer at Stack Overflow.</p>

<p>Previously:</p>

elected moderator on Stack Overflow and Programmers SE
community manager at Stack Overflow

<p>Email me a link to your favorite Wikipedia article: <code>adam@stackoverflow.com</code>.</p>
" Views="19" UpVotes="0" DownVotes="0" ProfileImageUrl="https://i.stack.imgur.com/WzDd8.jpg?s=128&g=1" Age="33" AccountId="37099" />

key	Id	Reputation	CreationDate	DisplayName	…	Age	AccountId
val	1	101	…	…	…	33	37099

Class TopTenMapper stores a map of user reputation to the record and use the TreeMap to store the processed input records. TreeMap is like a red-black tree which the minimun is stored at the root after each insertion.

Void map(Object key, Text value, Context context) transforms xml to map and skip over the rows that do not contain user data. In addition, if there is more than 10 users in repToRecordMap, we need to remove the unnecessary data as well.

Void cleanup outputs our ten records to the reducers with a null key. The cleanup method gets called once after all key-value paris have been through the map function.

Public static class TopTenReducer stores a map of user reputation to the record.

Void reduce(NullWritable key, Iterable<Text> values, Context context) pushes the results into Hbase.

Figure 2 illustrates the UML of topten.java.

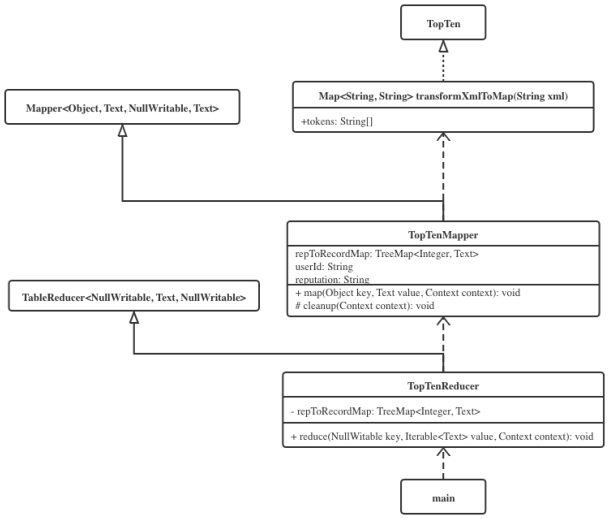

Figure 2 The UML of topten.java

3. Way of Running the Code:

According to the Lab 1 instruction page 14-15

(1) Start the HDFS NameNode and DataNode (if they are not running). Then create a folder input in HDFS, and upload the files in it.

(2) Start the HBase and the HBase shell.

(3) Create the HBase table topten with one column family info to store the id and reputation of users.

(4) Set the environment variables.

(5) Change directory to the src folder and make a target directory, topten classes, to keep the compiled files. Then, compile the code and make a final jar file.

(6) Run the application

(7) Check the result in the HBase shell

4. The Result of Code:

Becasue Hbase stores everything as bytes so we need to convert it to readable integer format in Hbase shell by using the command

org.apache.hadoop.hbase.util.Bytes.toInt("value(e.g. \x00\x00\x080).to_java_bytes").