HBase WordCount Project

Abderrezak Allalen

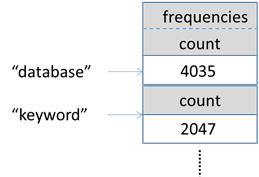
Varun Patil

Indiana University

March 29th 2015

**Project Introduction:**

The goal of this project is to implement HBase and Hadoop MapReduce to write an HBase WordCount program to count all unique term’s occurrences from the clueweb09 dataset. Each row of column family (“frequencies”) is unique, and the rowkey is the unique term stored in byte format, column name is “count” and value is term frequency. The result is then loaded to WordCountTable. Table 1 below shows HBase WordCountTable schema.

**Figure 1:** Hbase WordCountTable Schema

**HBase:**

It is an open source, distributed, column oriented, and stored map data source. HBase has the capacity to store large amount of data, thus it acts as a data source. The data is stored in tables and each table has multiple rows and fixed number of column families. Rows as sorted by row keys which are implemented as byte arrays.  
  
**Figure 2:** ClueWeb09Table Schema  


ClueWeb09 table is created in HBase after the working environment is set and ready. The input data is then loaded into ClueWeb09. Table 2 above shows HBase ClueWeb09 schema.  
The MapReduce functions would then use the data source in HBase to run the application.

**MapReduce Code**

**The Mapper Class:**

In the mapper class we were supposed to count words in each row of the data set and output <word, count> pair using the “getWordFreq” as show in the code below.  
*HashMap<String, Long> hash=getWordFreq(content);*The frequency (value) and word (Key) are initiated as follows: *LongWritable frequency = new LongWritable();  
Text word = new Text();*   
Finally looping through the hash map and outputting the <word(key), frequency(value)> pairs for each word in the content of Text using the following:  
*for (String key:hash.keySet()){  
word.set(key);  
frequency.set(hash.get(key));  
context.write(word,frequency);  
}*

**The Reducer Class:**

In the reducer class we were supposed to collect the <word, frequency> pairs output from the mapper and aggregate it to a single result. In this step the occurrences of each word are summed to pairs of <word, occurrence> using the following code:

*long totalFreq = 0;  
for (LongWritable val : freqs){  
totalFreq += val.get();   
}*  
Then the write the result back to HBase table using the put operation as shown below:  
*String key = word.toString();  
byte tFreq[] = Bytes.toBytes(totalFreq);  
byte wKey[] = Bytes.toBytes(key);  
Put put = new Put(wKey);  
put.add(Constants.CF\_FREQUENCIES.getBytes(), Constants.QUALIFIER\_COUNT.getBytes(), tFreq);*Finally, write out the output using *context.write(null,put);*

Along with this report, the source code and the output files are included in the zip file