# Project 4: HBase WordCount
# Cloud Computing
# Spring 2017

Professor Judy Qiu

## Goal

Write an HBase WordCount program to count all unique terms' occurrences from the clueWeb09 dataset. Each row record of columnfamily "frequencies" is unique; the rowkey is the unique term stored in byte format, column name is "count" and value is the term frequency shown in all documents. Load the result to HBase WordCountTable. Figure 1 shows the schema of WordCountTable. You will compare the results of your finished run to a correct version we will supply to you.

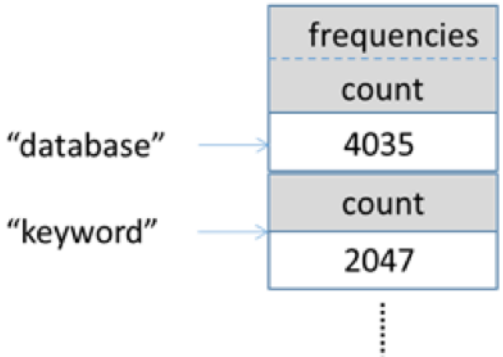


Figure 1: WordCount table schema for storing unique term's occurrences

## Deliverables

Zip your source code and report in a file named username_project4.zip

## Evaluation

The point total for this project is 1.5, where the distribution is as follows:

- Correctness of your code and output (1 points)
- Completeness of written report (0.5 points)
- The report should explain the logic behind your code.

## Prerequisites

You?ll need to load data to HBase before trying this assignment. Please follow **project4_Prereq.pdf** for more information.

# Introduction

WordCount is a simple program which counts the number of occurrences of each word in a given text input dataset. It fits very well with the map/reduce programming model, making WordCount a great example to understand the Hadoop MapReduce programming style. Instead of loading the data from HDFS, we will load our data directly from existing HBase records which store the similar content structures on HBase and HDFS.

In this homework and the next homework (Building an Inverted Index) we use the same source code, which can be found in: */root/MoocHomeworks/HBaseWordCount*.

## Clueweb09 dataset

We are using the ClueWeb09 dataset, which was created to support research on information retrieval and related human language technologies. It consists of about 1 billion webpages in ten languages that were collected in January and February 2009. The dataset is used by several tracks of the TREC conference[2]. Since the ClueWeb09 dataset is composed of webpages crawled from the Internet, the uploaded table schemas are designed as shown in Figure 2.


Figure 2: Data table schema for storing the ClueWeb09 dataset

So, while similar to Hadoop WordCount [3], the differences are that data is stored on HBase and URI is the "filename" that contains all the text content.

## Mapper, Reducer and Main Program

Now we are going to implement the HBase WordCount. Our implementation consists of three main parts:

- Mapper
- Reducer
- Main program

### Mapper

A Mapper overrides the map function from the Class "org.apache.hadoop.hbase.mapreduce.TableMapper<Text, LongWritable>" which provides <key, value> pairs as the input. A Mapper implementation may output <key, value> pairs using the provided Context. <key, value> of this map function is <rowkey, content>, where the key is the rowkey of an HBase record related to a specified URI, and the content is the stored text of that URI. Your Map task should output <word, frequency> for each word in the content of text.

*Pseudocode*

```
void Map (key, value){
    for each word x in the content of a hbase record:
    context.write(x, freq);
}
```

*Detailed implementation*

```
static class WcMapper extends TableMapper<Text, LongWritable> {
    @Override
    public void map(ImmutableBytesWritable row, Result result, Context context) throws
    IOException, InterruptedException {
      byte[] contentBytes = result.getValue(Constants.CF_DETAILS_BYTES, Constants.
    QUAL_CONTENT_BYTES);
      String content = Bytes.toString(contentBytes);

      // TODO: write your implementation for counting words in each row, and generating a <
    word, count> pair
      // Hint: use the "getWordFreq" function to count the frequencies of words in content

    }
}
```

### Reducer

A Reducer collects the intermediate <key, value> output from multiple map tasks and assembles a single result. Here, the reducer function will sum up the occurrence of each word to pairs as <word, occurrence >, then write it back to an HBase table with put operations which contain the key-value pair information of each word. *Pseudocode*

```
void Reduce (keyword, <list of value>){
    for each x in <list of value>:
        sum+=x;
        context.write(rowkey(x), freq);
}
```

*Detailed implementation*

```
public static class WcReducer extends TableReducer<Text, LongWritable,
    ImmutableBytesWritable> {
      @Override
    public void reduce(Text word, Iterable<LongWritable> freqs, Context context)
                throws IOException, InterruptedException {
        /*TODO: write your implementation for getting the final count of each word
        and putting it into the word count table
        Hint -- the schema of the WordCountTable is:
           rowkey: a word, column family: "frequencies",
           column name: "count", cell value: count of the word
        Check iu.pti.hbaseapp.Constants for the constant values to use.
  */
      long totalFreq = 0;
     }
}

```

### Main program

The main function has been provided as standard initialization, although you can modify it to suit your own style. Hint: the provided code is designed for using put operations in the reducer content.write() function. Before writing the codes, please read the HBase MapReduce tutorial first [4].

## Edit

The sketch code is stored within the provided VirtualBox image Environment Setup. You may use linux text editor vi/vim to add your code.

```
$ cd /root/MoocHomeworks/HBaseWordCount/
$ vim src/iu/pti/hbaseapp/clueweb09/WordCountClueWeb09.java

```

### Compile and run your code

For your convenience, we have provided a one-click script compileAndExecWordCount.sh for compiling and execution. Standard error messages such as "compile errors, execution errors, etc." will be redirected on the screen. You may debug it based on the returned messages.

```
$ cd /root/MoocHomeworks/HBaseWordCount
$ ./compileAndExecWordCount.sh
```

## View the result

The result is generated as /root/MoocHomeworks/HBaseWordCount/output/project1.txt.

```
$ cd /root/MoocHomeworks/HBaseWordCount
$ cat output/project1.txt
```

## References


[1] Clueweb09 dataset. `http://lemurproject.org/clueweb09/`.

[2] Hadoop WordCount. `http://salsahpc.indiana.edu/csci-b649-spring-2014/projects/project1.html`.

[3] HBase MapReduce Examples. `http://hbase.apache.org/book/mapreduce.example.html`.