MapReduce Programming Project for CMPT 741 – Version 3 November 4, 2014)

(Since this project is offered for the first time, there may be refinement based on the student feedback, in which case you will be notified of further version). This is a group-based project. Each group has two students and has one submission.

Objectives: find all frequent itemsets in TWO passes of databases using MapReduce programming.

1. Read the chapters 6.4.1, 6.4.3, and 6.4.4 of Resource 2 (Mining Massive Datasets). These chapters describe a parallel computation of frequent itemsets in TWO passes of the database with a MapReduce implementation. It has two phases. The **first phase** divides the large input data set into k sub-files such that each sub-file is small enough to be read into the memory entirely, therefore, you can use any "in-memory" algorithm for finding the frequent itemsets in this sub-file. The result is k sets of frequent itemsets, C1,C2,…,Ck, one set for each sub-file. The **second phase** takes the union of C1,C2,…,Ck as the set of candidates and finds those candidates that are actually frequent in the whole input data set.
2. **Implement the two pass MapReduce algorithm as described in 6.4.4**. For the "in-memory" algorithm of finding frequent itemset in the first Map function, you can find codes on internet or write your own codes such as the A-priori algorithm. But you have to implement all Map and Reduce functions. You need to run your programs on Hadoop machines described in the "Project Resources and Instruction" section below.
3. Evaluate the correctness of your implementation by comparing its result with the result produced by any single machine implementation. Initially you can use your own data sets, which are any sets of baskets, though we will provide sample data sets. For ease of evaluation, you assume the following input parameters and output format.

Input parameter:

File, k, s

Where File contains the input data set, k is the number of sub-files, and s is the support threshold in percentage.

Output format: first print the number of frequent itemsets on one line, then print all frequent itemsets, each on a different line, in the descending order of support. Break the tie by alphabetic order of itemsets. Here is an example of the output of five frequent itemsets, where the numbers in the brackets are the support.

5

milk (10)

beer (9)

dog (9)

diaper (6)

cat (5)

Project grading: report (80%) and demo (20%). The report should cover the important information about the project and implementation, including the description of project, high level approach to key steps, pseudo codes, sample test data and its results, discussion, and source in Appendix. The key steps include the pseudo codes for map and reduce functions (something likes those in notes), algorithms for finding frequent itemset in first phase, and the algorithm for counting candidates in second phases. Limit the report to 3000 words, excluding the actual code in Appendix. The evaluation criteria include clarity, organization, informativeness, readability, and correctness. Important: a *proper* level of details is essential for the marker to get the main ideas; too much details or too little details does not achieve this goal.

For the demo, you will be given a *new* testing data set, k and s values, and asked to output the results in the same format as above.

Deadline of report: Nov 20 2014 (Thursday). Demo dates will be announced at a later time.

Project Resources and Instruction

TA office hours: Chenyi Zhang, chenyiz@sfu.ca. Thu 10:00am-11:30am, ASB9838_TA_3. All programming related issues should be referred to the TA.

Basic Map-Reduce programming: Once you login the Hadoop account, you can run map and reduce functions. You can find programming related information at the Hadoop home page:

http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

For example, an example code of performing word count is given at

http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v2.0

Hadoop setup and connection: You have two options for Hadoop setup.

Option 1: Using SFU cluster. A Hadoop account has been created for each 741 student under the SFU ID. You can use the following step to connect to the account.

(1) Connect to head node

```
ssh <user_id>@hadoop.rcg.sfu.ca
```

if you work outside SFU network (e.g., at home), first try

```
ssh <user_id>@oak.fas.sfu.ca
```

then connect by step (1).

Option 2: You may also set up a single node cluster on your machine such as laptop.

(1) Follow this instruction for Linux and Mac user:

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html

Recommend to set Pseudo-Distributed mode and YARN.

(2) Following this instruction for Windows user:

http://www.srccodes.com/p/article/38/build-install-configure-run-apache-hadoop-2.2.0-microsoft-windows-os
http://www.codeproject.com/Articles/757934/Apache-Hadoop-for-Windows-Platform

In both options, once set up, you can work with HDFS via the commands available under Hadoop, such as

```
hadoop fs -<shell_command>
```

```
hadoop fs -ls /user/<user_id>
hadoop fs -copyFromLocal <local_path> <HDFS_path>
hadoop fs -rm <HDFS_path>
```

Refer to the resources under **Basic Map-Reduce programming** for more commands such as running a mapreduce program. Note that you only can access /user/<your user_id> the HDFS (Hadoop distributed file system).