**CS 585 Project 1**

**Total Points: 120**

**Release Date**: **02/03/2015**

**Due Date: 02/13/2015 (11:59 PM--Midnight)**

**Team member: Changtao Wen, Junwei Guan**

1. Creating datasets

The java code can be run on local machine (windows) or in virtual machine (Linux).

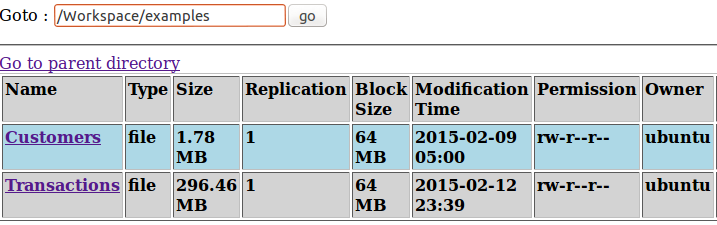
For running in local machine: the Customer.txt is stored under “d:\\Customers.txt”

For running in virtual machine: file stored under "/home/ubuntu/Workspace/Customers.txt"

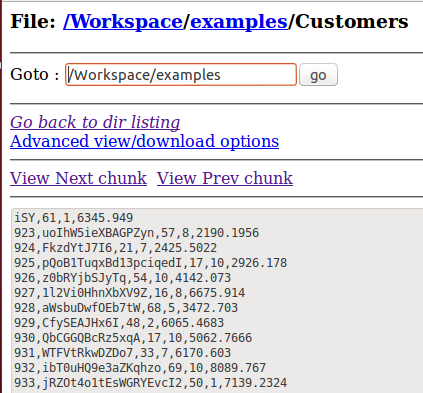
As well as the Transaction.txt

Ps: you need to annotation corresponding line to change the storage formats, our code is for local machine in default.

1. Uploading Data into Hadoop

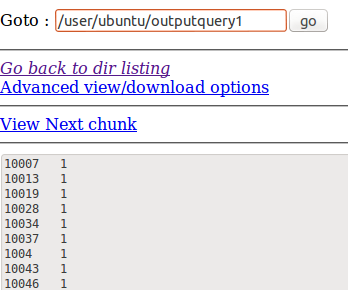


We uploaded these two documents to this direction in HDFS: /Workspace/examples



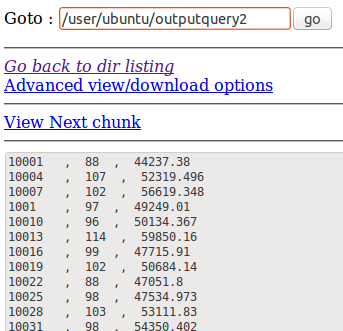
As we can see, the files are divided into blocks, and each block might not contain whole information of one tuple.

1. Writing MapReduce Jobs
   1. Query1



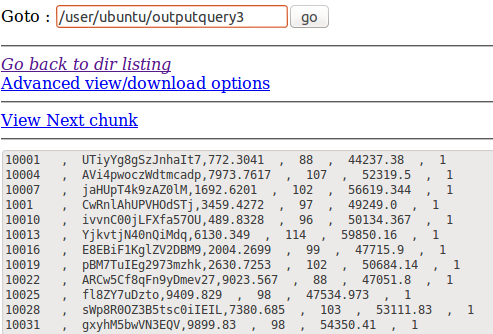
The customer whose Country Code between 2 and 6(inclusive) are listed on the left column, right column is the times such id appear.

* 1. Query2

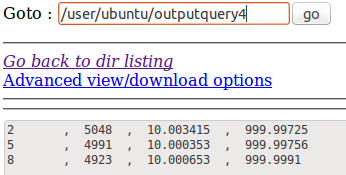


When running with a combiner, the time for the job is clearly decreased. Because when running with combiner, we do not have to wait for maps to complete all their jobs, instead, we can have combiners doing parts of reducers job when some of the mappers have given certain outputs. In a word, combiners can help reducers do some work and it can start to work even when mappers have not finished which can save large amount of time in certain circumstance.

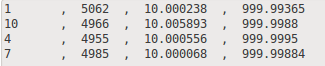
* 1. Query3



* 1. Query4







Below are the outputs of three reducers respectively.

Take query4 for example, the code to make jar and run the mapreduce is as follows:







Note that there are two input path for query4, since DistributedCache.addCacheFile() are used to put the Customer.txt into main-memory to do the job in just one mapreduce.