## HU Extension Assignment 08 E185 Big Data Analytics

### Handed out: 04/05/2013 Due by 5:30PM on Friday, 04/12/2013

**Problem 1)** Rewrite attached InverterCounter.java class in new MapReduce API. Demonstrate that new class generates the same values as the class written in old API.

While testing new class demonstrate that you can output intermediate results only and not run any reducers.

**Problem 2)** Imagine that you are a Linux person and you have not a single machine with an Eclipse. Your boss is adamnt and wants a graphical histogram of citation counts you are calculating with a class like the attached CitationHistogram.java class. Write a new Hadoop program that will print a true (graphical) histogram of log10(citation\_count) values using a string of asterixes (\*) to indicate the value. You histogram will look approximately like this:

1-20 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

21-40 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

41-60 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

. . . .

760-780 \*\*\*\*\*\*

Use buckes of size 20.

**Problem 3)** Attached set of slides under the title “MapReduce calculation of Pi.ppt” It shows to you how you could calculate the value of number Pi (π) by randomly generating points in a two dimensional square. The ratio of points that fall into the circle and the number of points that fall into the square appears to be equal (close) to . Try to formulate this approach as a MapReduce job. Experiment also with the number of mappers and reducers your job will use. Class JobConf has methods setNumMapTasks() and setNumReduceTasks(). Use them. Report on your findings. You are not expected to calculate to the precision of 1000 decimal places. You just want to do a better job than you would do with a single random number generator.

**Problem 4)** We can execute the two MapReduce jobs manually one after the other, the way we did it in the class with InverterCounter job, the outputof which we used as the input to CitationHistogram job. It would be more convenient to automate the execution sequence. You can chain two or more MapReduce jobs to run sequentially, with the output of one MapReduce job being the input to the next. Chaining MapReduce jobs is analogous to Unix pipes .

mapreduce-1 | mapreduce-2 | mapreduce-3 | ...

Chaining MapReduce jobs sequentially is quite straightforward. Recall that a driver sets up a JobConf object with the configuration parameters for a MapReduce job and passes the JobConf object to JobClient.runJob() to start the job. As JobClient.runJob() blocks until the end of a job, chaining MapReduce jobs involves calling the driver of one MapReduce job after another. The driver at each job will have to create a new JobConf object and set its input path to be the output path of the previous job. You can delete the intermediate data generated at each step of the chain or at the time when they are not needed any more, at the end. You can perform the file deletion with command like this

FileSystem.delete(Path f, boolean recursive);

You can control input and output file locations (paths) programmatically using Path class.

Path in = new Path("HDFSDirectory/filename");

Path first\_out = new Path("hdfs\_directory”);

Demonstrate that your chained MapReduce job for calculating Citation Histogram starting with patents citation file cite75\_99.tx will produce the same result as the sequence of two jobs used in class.

**Submission Note:** Please capture all the steps of your implementation in an MS Word document. Please add comments indicating what is it you are accomplishing with every step. Please submit a copy of working code.

Please place all files you want to submit in a folder named: HW08. Compress that folder into an archive named E185\_LastNameFirstNameHW08. ZIP. Upload the archive to the course drop box on the class web site. Please send comments and questions to [cscie185@fas.harvard.edu](mailto:cscie185@fas.harvard.edu)