**Virginia Tech.** **CS 4604 – Introduction to DBMS**
**Computer Science** **Spring 2016, Prakash**

# Homework 5: MapReduce
## (due March 28th, 2016, 4:00pm, hard-copy in-class)

*Reminders*:

a. Out of 100 points. Contains 4 pages.
b. Rough time estimates 6-8 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Lead TA for his homework: Sorour Amiri.

## Q1: AWS MapReduce [100 points + 10 points *bonus*]

In this problem, we will learn how to use Hadoop to generate degree distributions of a large graph/network. The idea is to convince you that using Hadoop on AWS has now really become a low-cost/effort proposition (compared to setting up your own cluster).

Familiarize yourself with AWS (Amazon Web Services). Read the set-up guidelines posted on piazza---to set up your AWS account and redeem your free credit ($35). **Do the setup early!**

**Link to setup:**
**http://courses.cs.vt.edu/~cs4604/Spring16/homeworks/hw5/AWS-setup.pdf**

**Link to 'how to run a sample wordcount application on AWS':**
**http://courses.cs.vt.edu/~cs4604/Spring16/homeworks/hw5/SBS-AWS-Wordcount.pdf**

The pricing for various services provided by AWS can be found at http://aws.amazon.com/pricing/. The services we would be primarily using for this assignment are the Amazon S3 storage, the Amazon Elastic Cloud Computing (EC2) virtual servers in the cloud and the Amazon Elastic MapReduce (EMR) managed Hadoop framework. Play around with AWS and try to create MapReduce job flows (not required, or graded) or try the sample job flows on AWS. Of course, after you are done with the HW, feel free to use your remaining credits for any other fun computations/applications you may have in mind! These credits are applicable more generally for AWS as a whole, not just MapReduce.

The questions in this assignment will ideally use up only a **very small fraction of your $35 credit**. AWS allows you to use up to 20 instances total (that means 1 master instance and up to 19 core instances) without filling out a "limit request form". For this assignment, **you should not exceed this quota of 20 instances**. You can learn about these instance types by going through the extensive AWS documentations.

**Important:** You should always check how much credit you have left from time to time. Sometimes this takes a while to update. Always make sure to test your mappers and reducers on some sample local data before using the AWS. Note that if you run over, your card will be charged! Ideally for this assignment you would need a small fraction of your credits. To check how much credit you have spent go to "Billing and Cost Management Dashboard" from the AWS management console. The link to this page is in the dropdown under your name on the top right corner of the AWS management console. On this page, you can check the "month-to-date" credit you spent. You should also check the "Bills" page (sometimes that is updated more frequently than the credit page). The link to this page is in the left column of the "Billing and Cost Management Dashboard" page. Click on the "Bills" link.

We will use data from http://www.livejournal.com: Live Journal is a free on-line community with almost 10 million members; a significant fraction of these members are highly active. (For example, roughly 300,000 update their content in any given 24-hour period.) LiveJournal allows members to maintain journals, individual and group blogs, and for people to declare which other members are their friends. For purposes of this assignment, we will assume that it is an *undirected* social network i.e. if A has declared B as a friend, we will assume B has also declared A as a friend, and hence there is an undirected edge between nodes A and B.

We have uploaded the undirected LiveJournal (LJ) friendship network to the following Amazon S3 bucket (directory):

s3n://cs4604-2016-vt-cs-data/livejournal/

Refer to the guidelines to see how to set this data as input to your MapReduce job. We have provided a screenshot to configure the EMR cluster, which demonstrates how to access the input data from the given bucket.

We have stored the graph as an edge-list. The format of the data is:

*Id* *FriendID* (tab separated)

For each node *Id* we store all friends of that user. This means that there is an edge between Node #*Id* and Node #*FriendID*. A data snippet will look like:

```
0	1
0	2
0	3
0	7
1	0
1	7
. . . . . .
```

Note that data is repeated: e.g. if node 0 is a friend of node 1, then node 1 is also a friend of node 0 (hence we will have both 0 1 and 1 0 as rows in the data). Both 0 1 and 1 0 denote the same edge (between 0 and 1) though. We have chosen to store the data like this for convenience.

We want to compute the degree distribution of the LJ graph. We will explain what we mean by it step-by-step. While this is not too hard a task in general, we want to use Hadoop/MapReduce because of the large size of the graph. You can run your code using the AWS console by following the instructions in AWS-Setup (the easier way). You are also welcome to use the elastic-mapreduce command-line interface based on Ruby (read the responding part in AWS-Setup document).

For all the questions below, feel free to use the fact that you are given $N = 4847571$ for the Live Journal graph.

Q1.1. (20 points) Imagine all the data was stored in a relational database table called `Friends` (Id, FriendId). Write a *single* (possibly nested) SQL query that operates on `Friends` and returns a relation `Counts` (NumFriends, NumIds). If this relation stores the tuple (k, l), then it means that there are l distinct users (Id values) in `Friends`, each of who has exactly k friends. As you can imagine, you do not know beforehand all the different values of k (or l) that should appear in `Counts`. Hence your SQL query should be able to figure out all these values automatically and correctly. (Note: We do not want you to run anything or work with the actual data for this question; just write down the SQL query).

Now let us define $P(k) = \frac{l}{N}$, for each value of *k*, *l* you would get in `Counts`. *P(k)* is exactly the degree distribution of the associated social network, as it tells us how the degrees are 'distributed' in the given graph. Note that *P(k)* is a valid probability distribution i.e. $\sum_k P(k) = 1$.

Q1.2. (5 points) Suppose you plot the values with *P(k)* on the y-axis and *k* on the x-axis. Before computing the actual empirical degree distribution *P(k)*, write down your expectations of this plot i.e. what probability distribution do you expect to find for the degree distribution of this LJ social network (e.g. Gaussian? Or Exponential? Or Poisson? etc.)? Explain in 1-2 lines.

Q1.3. (70 points) Write the mapper and reducer in Python to calculate the degree distribution $P(k)$ for the LJ graph. Run it on AWS EMR and copy the output to your local machine and then plot $P(k)$ *(y-axis)* vs *k (x-axis)* in both *linear* and *log-log* scales.

Q1.4. (5 points) Do your answers in Q1.2 match with what you actually got in Q1.3? If not, did any of your prior assumptions fail? (Explain in 1-2 lines).

Q1.5. (BONUS 10 points) Consider the log-log plot of *P(k)* you found in Q1.3. Is there any analytical function y = f(x) that will 'fit' this empirical distribution? Write down the functional form f(x), and show the fit (i.e. plot the function over your empirical distribution). You may have to manually guess and set the 'correct' parameters in the function to match the empirical distribution approximately e.g. the general functional form of a line is y = mx + c, so if you

want to fit a line to some given data, you need to guess some suitable m and c. Feel free to search the web for this question, about how to fit lines/curves etc.

**Some Important Notes/Hints**:

1. You will need two consecutive MapReduce jobs for Q1.3. Your SQL query in Q1.1 might be useful for giving you an idea.

2. Again, feel free to use the fact that you are given $N = 4847571$. Hence first just get the *number of* nodes with degree $k$ (i.e. $l$ for each $k$) using AWS, and then simply divide everything by $N$ while plotting on your local machine.

3. We have also created a cs4604-hw5-help.zip file available from here: http://courses.cs.vt.edu/~cs4604/Spring16/homeworks/hw5/cs4604-hw5-help.zip. This file contains code/files that you may find useful. Make sure you test your code on the sample dataset we have in the zip file (first locally, then on AWS), before you run it on the full LJ dataset (to avoid unnecessary extra charges to your account).

4. You can check and monitor your usage under the billing section of your account. Check the AWS setup document for more details.

5. The deliverables for Q1.3 include mapper-1.py, mapper-2.py, reducer-1.py, reducer-2.py, degree-dist.out (the raw final output from AWS) and degree-dist.jpg (the plot). Zip all of these as YOUR-LASTNAME.zip. Then email Sorour (esorour@vt.edu) the zip file with the subject "CS4604:HW5-Code-Q1".

6. **Also include all the code/plots in the hard-copy (everything except the raw .out files).**